MASTER EN SCIENCES INFORMATIQUES

Automatisation de la vérification d'une base de connaissances: apllication à l'interface Expert de CosmicXpert

Duterme, Christophe; Fabry, Nicolas

Award date:
2004

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique.

Automatisation de la vérification d'une base de connaissances. Application à l'interface « expert » de CosmicXpert.

par

Christophe DUTERME & Nicolas FABRY

Mémoire présenté pour l'obtention du diplôme de

Maître en informatique

Année académique 2003-2004

Résumé

A l'heure actuelle, une part importante du budget de toute organisation est allouée en logiciel, devenu un outil incontournable. Afin de gérer ces dépenses en logiciel et de minimiser les risques de développement et de maintenance de ceux-ci, des mesures et des modèles utilisant celles-ci sont nécessaires. L'une des mesures existant aujourd'hui sur le marché, est la mesure fonctionnelle COSMIC-FFP. Le système à base de connaissances CosmicXpert a été développé, sous l'impulsion du professeur Jean-Marc Desharnais, pour faciliter et améliorer l'apprentissage de cette mesure et les performances du mesureur.

La théorie de la mesure fonctionnelle étant en évolution, il est indispensable de faire transparaître ces changements dans la base de connaissances du système.

Actuellement, la littérature ne propose pas de processus permettant la modification d'une base de connaissances hybride, tout en assurant sa vérification. Ce travail a pour objectif de proposer une solution permettant la modification d'une base de connaissances tout en la vérifiant automatiquement. Le mécanisme utilisé se décompose en deux niveaux : l'un issu de concepts du monde des bases de données, l'autre concernant le formalisme de conception.

Abstract

At the present time, an important part of the budget of each company is allocated to software. To manage these expenses and to reduce development and maintenance risks, measures and models using these measures are required. One of these measures currently on the market is the COSMIC-FFP functional size measurement method. The knowledge based system CosmicXpert has been developed, at the instigation of Professor Jean-Marc Desharnais, to help learning this method and increase the measurer's performances.

As the measurement theory evolves, the knowledge base of the system has to change.

Nowadays, literature doesn't offer any process to modify an hybrid knowledge base, verifying it at the same time. We wrote this thesis with the aim of building a solution that allows the modification of a knowledge base and that verifies it automatically. The used mechanism is made up of two different layers : one emerging from the Data Base world, the other regarding the conception's formalism.

Avant-propos

Le travail présenté dans ce document est basé sur le stage réalisé à l'ÉTS (École de Technologie Supérieure) de Montréal, Canada. La rédaction de ce document n'est pas seulement l'œuvre de deux étudiants, c'est pourquoi nous voudrions remercier toutes les personnes qui, de prêt ou de loin, ont permis la réalisation de ce travail.

Nous remercions tout d'abord le Professeur Habra, promoteur de ce mémoire, pour son suivi de la rédaction de ce document.

Nous remercions également le Professeur Desharnais, notre maître de stage et initiateur du projet CosmicXpert, pour son aide précieuse et ses bons conseils. Nous le remercions également pour son attention particulière à notre égard.

Nous pensons aussi à nos prédécesseurs sur le projet CosmicXpert, Tim Küssing, Julien Vilz et François Gruselin, sans qui ce travail n'aurait été possible.

Nous joignons enfin à nos remerciements nos parents et petites amies, pour leur soutien tout au long de nos études.

Table des matières

Table des illustrations et des tableaux

Introduction

Ce document présente un mécanisme de vérification automatique d'une base de connaissances. Pour ce faire, nous nous appuyons sur des travaux réalisés dans le cadre de la norme ISO [1] et sur la thèse de doctorat du professeur J.-M. Desharnais [2].
Cette thèse est à la base du projet CosmicXpert, qui tente de vérifier l'hypothèse selon laquelle un système à base de connaissances peut aider le personnel de la mesure à acquérir et à maintenir les connaissances nécessaires à la mesure fonctionnelle des logiciels.
Le premier prototype, développé en 2001 par T. Küssing [3], essaie de répondre à ces besoins. En 2002, afin d'éliminer la redondance trop présente dans le premier prototype, F. Gruselin et J. Vilz [4] ont tenté d'introduire une vérification et une validation du système, en formalisant les connaissances. Dans cette deuxième version de CosmicXpert, seuls les besoins des mesureurs ont été pris en compte. Elle ne permet plus aux experts de maintenir la base de connaissances.
Notre tâche est de compléter ce prototype en permettant aux experts d'introduire, grâce à une interface, des changements dans la base de connaissances. Cette procédure ne serait pas complète sans un mécanisme de vérification automatique, assurant le maintien de l'intégrité de la base de connaissances.

Dans un premier temps, nous tentons de définir les concepts utiles à la suite de notre raisonnement. Le chapitre 1 est consacré à la présentation des systèmes experts et plus particulièrement des systèmes à base de connaissances. Le second chapitre insiste sur la nécessité d'une phase de vérification et validation dans de tels systèmes. Le troisième ajoute une dimension supplémentaire à la problématique, par l'étude du concept d'intégrité présent dans le monde des bases de données.

La deuxième partie présente, au travers des chapitres 4, 5 et 6, les trois différents prototypes de CosmicXpert depuis la création du projet.

Dans une troisième partie, et plus particulièrement dans le chapitre 7, nous reprenons la méthodologie suivie, ainsi que les différents types de mécanismes utilisés lors de l'implémentation de notre prototype.
Enfin, avant de conclure, nous illustrons ce troisième prototype dans le chapitre 8, à travers le cheminement d'un expert désireux d'introduire un nouveau cas problème dans la base de connaissances.

Des annexes concernant le formalisme de conception, les fonctionnalités du système ainsi qu'un exemple d'utilisation clôtureront cet ouvrage.

Chapitre 1 : Concepts de base de connaissances

Avant d'entrer dans le vif du sujet, et de commencer à expliquer les concepts de vérification et de validation d'une base de connaissances, il semble utile de rappeler quelques concepts, notamment ceux de système expert et de base de connaissances.
Il est à noter que les théories existantes dans les domaines des systèmes experts, des bases de connaissances, ainsi que dans le domaine de leur vérification, sont en pleine évolution. Nous ne pouvons donc pas dresser une définition pour chaque concept que nous abordons. Nous tenterons néanmoins de convenir de définitions, de façon à pouvoir poursuivre sur des bases communes.

1.1 Concept de système expert

1.1.1 Définition

Les définitions d'un système expert peuvent varier. Certaines sont basées sur la structure, d'autres sur les composants fonctionnels, d'autres encore supposent le raisonnement à partir de règles. Voici quelques exemples, tirés de [5] :

> *« La connaissance des experts est souvent rare et précieuse. Les systèmes experts sont des programmes d'ordinateurs qui capturent une partie de cette connaissance et permettent la diffusion de celle-ci à d'autres. »*

> *« Un système expert (sur la base de connaissances) est un système de résolution de problèmes et d'aide à la décision basé sur des connaissances d'un domaine particulier et sur des règles logiques ou sur des procédures pour utiliser ces connaissances. Les connaissances comme la logique sont toutes deux issues de l'expertise d'un spécialiste du domaine.»*

> *« Un système expert est un programme qui émule l'interaction qu'un utilisateur pourrait avoir avec expert humain, pour résoudre un problème. L'utilisateur fournit*

les données. Le programme va poser des questions jusqu'à ce qu'il arrive à une conclusion. La conclusion peut être une solution unique ou une liste de solutions possibles. Le système peut expliquer en langage naturel comment il est arrivé à cette conclusion et pourquoi. »

De façon générale, on peut définir un système expert comme étant : « *un programme d'ordinateur qui réalise une tâche difficile généralement réalisée par un expert humain, sur la base de connaissances et de raisonnements.* » [5]

Dans le but de préciser la définition donnée, nous pouvons proposer une classification des systèmes experts en fonction des problèmes qu'ils résolvent. Ces catégories, tirées de [6], sont reprisent dans le Tableau 1.1.

Tableau 1.1 *Catégories génériques des applications d'ingénierie de la connaissance.*

Catégorie	Problème visé
Interprétation	Dégager des conclusions de haut niveau à partir de données brutes.
Prédiction	Induire des conséquences probables de situations données.
Diagnostique	Déterminer les causes de disfonctionnements d'un système à partir de symptômes observables.
Conception	Trouver une configuration de composants d'un système afin d'atteindre par exemple des objectifs de performance tout en satisfaisant un ensemble de contrainte d'exécution.
Planification	Concevoir une séquence d'actions visant à atteindre un ensemble d'objectifs en fonction des conditions de départ et des contraintes d'exécution.
Surveillance	Comparer le comportement observé d'un système par rapport à son comportement attendu.
Déboguer	Prescrire des remèdes pour les disfonctionnements.
Réparation	Exécuter un plan pour administrer un remède prescrit.
Education	Diagnostiquer, déboguer et réparer le comportement d'apprentissage.
Contrôle	Interpréter, prédire, réparer et surveiller les comportements du système.

1.1.2 Processus de développement d'un système expert

Le processus de développement d'un système expert a été défini par Waterman dans [7]. Ce processus est souvent appelé ingénierie de la connaissance (*knowledge engineering*). Il comprend une interaction entre le constructeur du système expert, appelé l'ingénieur de la connaissance (*knowledge engineer*), et un ou plusieurs experts humains du domaine concerné. L'ingénieur de la connaissance doit dans un premier temps, « extraire » des experts humains leurs procédures, leurs stratégies et autres règles établies pour la résolution de problèmes. Il doit ensuite construire la connaissance du système expert, comme montré dans la Figure 1.1 tirée de [7].

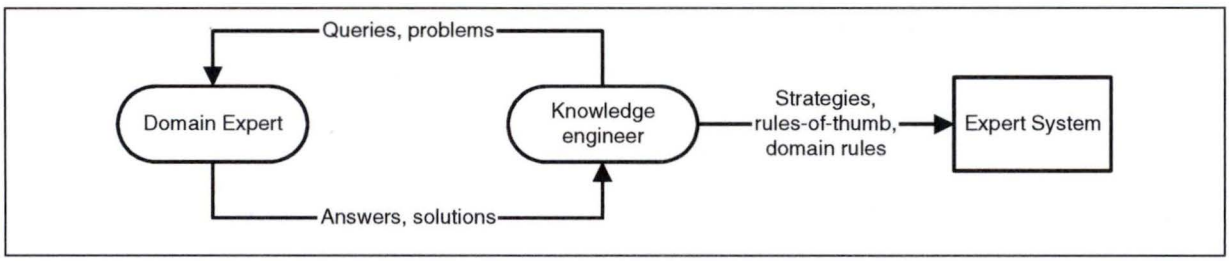


Figure 1.1 *Ingénierie de la connaissance : transférer la connaissance d'un expert à un programme.*

1.1.3 Les acteurs impliqués dans un système expert

Nous pouvons également tenter de cerner le concept de système expert par la voie actantielle. Waterman [7] considère que les principaux acteurs d'un système expert sont le *système expert*, l'*expert du domaine*, l'*ingénieur de la connaissance*, l'*outil de construction du système expert* et l'*utilisateur*. Leurs rôles et leurs relations sont résumés dans la Figure 1.2.

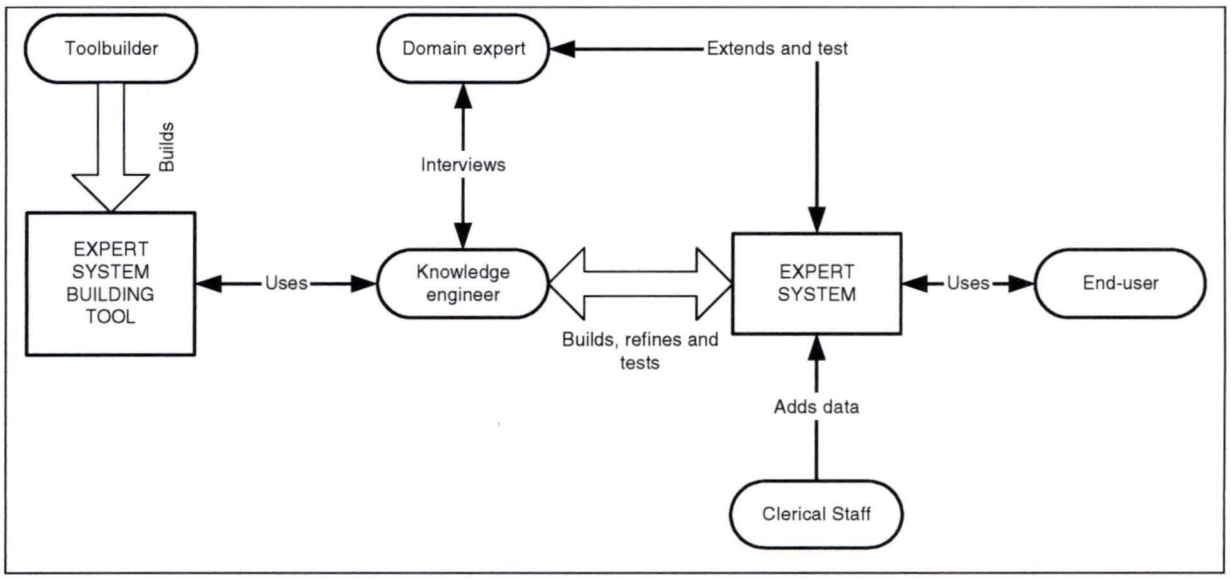


Figure 1.2 *Les acteurs d'un système expert.*

Le système expert est une collection de programmes ou de logiciels qui résolvent des problèmes dans un domaine particulier. On l'appelle *système* plutôt que simplement *programme* parce qu'il contient un composant pour la résolution du problème ainsi qu'un composant de support (aide pour l'utilisateur).

L'expert du domaine est une personne réputée pour ces connaissances et pour sa capacité à produire de bonnes solutions à des problèmes dans un domaine particulier. L'expert utilise des astuces et des raccourcis pour chercher une solution de façon plus efficace et le système expert modélise ces stratégies.

L'ingénieur de la connaissance interroge les experts, organise la connaissance, décide d'une façon de représenter cette connaissance dans le système et peut aider les programmeurs à écrire le code.

L'outil de construction du système expert est le langage de programmation utilisé par l'ingénieur de la connaissance et par les programmeurs pour construire le système expert.

L'utilisateur est l'humain qui utilise le système une fois qu'il a été développé.

1.2 Concept de connaissance

Dans la suite de ce document, nous parlerons plus généralement de *système à base de connaissances* (SBC), au lieu de système expert. Il existe une différence importante entre ces deux concepts. Tel que mentionné dans la section précédente, on peut dire qu'un système

expert a comme objectif de remplacer un expert humain, en s'appuyant sur des connaissances d'un domaine. Un système à base de connaissances utilise également des connaissances mais il n'a pas la prétention de remplacer un expert humain. Il s'agit d'une définition plus modeste, de plus en plus répandue dans la littérature. Communément, on peut dire que tout système expert est un système à base de connaissances mais tout système à base de connaissances n'est pas un système expert.

Maintenant que nous avons défini un système expert et un système à base de connaissances, nous pouvons nous intéresser à la façon dont la connaissance est représentée à l'intérieur du système, de sorte que celle-ci puisse être utilisée.

Le terme connaissance est utilisé en Intelligence Artificielle pour parler des informations dont a besoin un programme pour pouvoir se comporter de façon intelligente. Ces connaissances peuvent prendre plusieurs formes. On parle dans le cas des SBC de plusieurs représentations.

1.2.1 Choix d'une représentation

Une représentation a été définie par Winston cité dans [8] comme étant « *un ensemble de conventions syntaxiques et sémantiques qui rendent possible la description de choses* ». Dans le cadre des SBC, la représentation de la connaissance suppose des manières systématiques particulières de codifier ce que connaît un expert dans un domaine précis. Une représentation de la connaissance doit rendre cette connaissance accessible et facile à utiliser via des mécanismes plus ou moins naturels.

Les principaux critères d'évaluation d'une représentation sont l'*adéquation logique*, la *puissance heuristique* et la *commodité notationnelle*.
L'adéquation logique signifie qu'on doit être capable d'exprimer la connaissance que l'on veut représenter. Nous verrons plus tard que certaines représentations sont plus ou moins efficaces pour exprimer l'un ou l'autre types de connaissances.
La puissance heuristique signifie que de la même façon que nous avons un langage expressif avec une syntaxe et une sémantique bien définie, il doit y avoir des moyens d'utiliser les représentations construites d'une façon particulière et de les interpréter pour résoudre les problèmes. De manière générale plus le langage est expressif, en terme de nombre de

distinctions sémantiques possibles, plus il est difficile de contrôler le schéma des heuristiques[1] pendant la résolution d'un problème.
La commodité notationnelle est une vertu car la plupart des SBC nécessite l'encodage de grandes quantités de connaissances, et cette tâche ne serait pas enviable si le langage de représentation était trop compliqué. Les expressions devraient être relativement faciles à écrire et à lire, et il devrait être possible de comprendre leurs significations sans savoir comment le système va les interpréter.

1.2.2 Les différents types de systèmes à base de connaissances

La connaissance des SBC est organisée de façon à séparer les connaissances relatives au domaine d'expertise et les connaissances relatives à la résolution du problème ou à la communication avec l'utilisateur. La collection des connaissances propres au domaine d'expertise est appelée la *base de connaissances* du système, alors que les connaissances propres à la résolution du problème sont appelées le *moteur d'inférence*.
Nous nous intéressons ici à la base de connaissances.

Dans la section 1.1 présentée ci-dessus, nous avons donné une classification des systèmes experts sur base des problèmes qu'ils résolvent. Cette classification répond aux catégories données par Waterman. Nous allons ici développer une classification des SBC sur base de la représentation de leurs connaissances.

1.2.2.1 Système basé sur des règles

Les systèmes basés sur des règles représentent les connaissances sous la forme d'expression « si ... alors ... ». Dans la littérature, ces règles sont souvent appelées règles « conditions-actions » ou « situations-actions » pour des raisons évidentes. Un exemple de règle est donné à la Figure 1.3. Leur principale utilisation réside dans l'encodage d'associations empiriques entre des données introduites dans le système et les actions que ce dernier doit exécuter en conséquence.
Un système basé sur des règles consiste en un ensemble de règles, un interpréteur de règles qui décide quelle règle il faut appliquer, quand et comment l'appliquer, ainsi qu'une mémoire de travail qui peut contenir des données, des objectifs ou des résultats intermédiaires.

---

[1] Le terme « heuristique » correspond, dans le cadre des systèmes experts, aux règles établies ou aux simplifications qui limitent efficacement la recherche de solutions.

```
if P1 & ... & Pn,
then Q1 & ... & Qm.
```

Figure 1.3 *Exemple de règle.*

1.2.2.2 Système basé sur un modèle

> *« Un raisonneur de base de connaissances dont l'analyse est basée directement sur les spécifications et les fonctionnalités d'un système physique est appelé système basé sur un modèle. Dans sa conception et utilisation, un raisonneur basé sur un modèle crée un logiciel de simulation »* [9].

La base de connaissances doit contenir selon Luger [9] :

- Une description de chaque composante de l'appareil à simuler.
- Une description de la structure interne de l'appareil, c'est à dire une représentation de chacun de ses sous-composants et leurs interconnexions.
- Des informations concernant les performances actuelles de l'appareil.

La représentation des connaissances d'un système utilisant un raisonneur de ce type peut prendre des formes très différentes.

Les relations entre composants et leurs fonctionnalités peuvent être décrites sous forme de règles. Le paradigme de programmation orienté objet est également un bon moyen de représenter les composants en terme d'objets et de méthodes.

1.2.2.3 Système basé sur des cas

> *« Une autre stratégie puissante utilisée est le raisonnement par cas. Un cas est constitué d'un exemple ou d'un problème passé et de ses différentes solutions. Le raisonnement par cas (CBR: Case-based reasoning) utilise une base de donnée explicite de solutions de problèmes à transposer sur des situations de résolutions de problèmes nouveaux. Ces solutions proviennent d'expert humain à travers un processus d'ingénierie de connaissance ou proviennent de résultats de succès ou d'échec précédant du système »* [9].

Kolodner [10] décrit le cycle de raisonnement basé sur des cas à l'aide de la Figure 1.4.

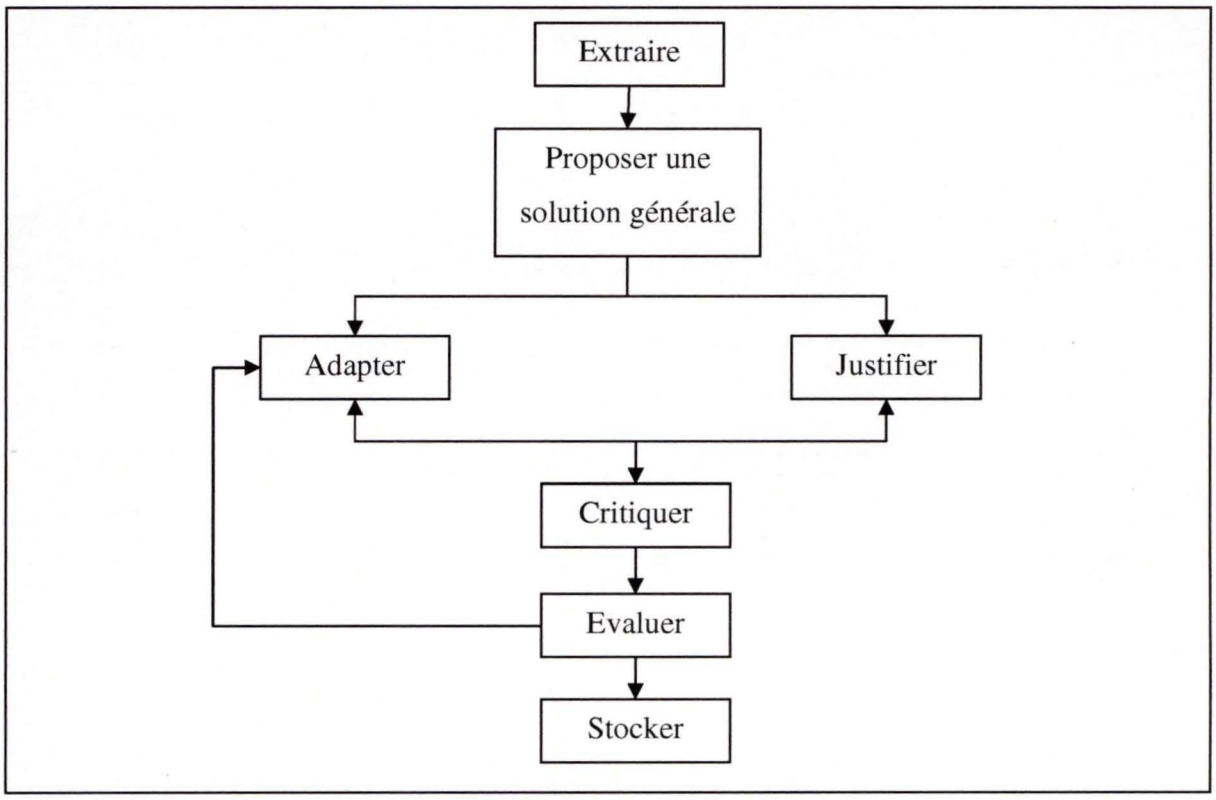


Figure 1.4 *Le cycle de raisonnement basé sur des cas*

L'*extraction d'un cas* se fait souvent en deux phases :

- Appel à des anciens cas : l'objectif de cette phase est de sélectionner un ensemble de cas qui pourrait potentiellement aider l'utilisateur dans son raisonnement pour solutionner le nouveau cas.
- Sélection du meilleur sous-ensemble : l'objectif de cette phase est de réduire l'ensemble des cas générés dans la première phase à un ou plusieurs cas les plus riches.

L'étape de *proposition d'une solution générale* consiste à extraire des cas sélectionnés dans l'étape précédente, les morceaux nécessaires pour construire une solution générale au nouveau cas.

L'étape d'*adaptation* consiste à adapter la solution générale proposée à l'étape précédente pour résoudre le nouveau cas.

Avant d'essayer la solution sur le cas spécifique, elle est *justifiée*. En général, toutes les données pour valider la solution ne sont pas disponibles.

La solution est *critiquée* en la comparant avec d'autre solution de cas présent dans la base des cas. Ces solutions nécessitent également une adaptation.

L'*évaluation* est la phase la plus importante du cycle de raisonnement par cas. C'est dans cette phase que le système reçoit les informations concernant la confrontation de sa solution avec le monde réel.

L'utilisateur analyse la solution en fonction du cas spécifique qu'il veut résoudre. Il retourne ensuite son analyse au SBC qui peut de cette façon apprendre c'est-à-dire augmenter sa base de connaissances pour ce type de problèmes et éviter certaines erreurs dans le futur.

Le cas spécifique une fois solutionné est alors enregistré dans la mémoire du SBC. Cette phase est appelée *stockage*. Le cas doit comprendre l'énoncé du thème, la solution et toutes les informations qui permettent au système de l'utiliser.

1.2.2.4 Les systèmes hybrides

> *« Un domaine de recherche et d'application important est la combinaison des différents modèles de raisonnement. Avec une architecture hybride, deux paradigmes ou plus sont intégrés pour obtenir un effet de coopération dans lequel les forces de l'un compense les faiblesses de l'autre »* [9].

Comme le décrit Luger, les systèmes hybrides sont constitués par l'application de plusieurs techniques de représentation de la connaissance. Ce choix de représentation est utilisé afin de compenser les faiblesses d'un modèle par les forces d'un autre modèle, en vue d'obtenir le maximum d'efficacité du système.

Chapitre 2 : Vérification et validation des systèmes à base de connaissances

2.1 Définition du processus de vérification et validation

Avant toute chose, et dû à la confusion habituelle entre la vérification et la validation, il est bon de rappeler brièvement leurs définitions :

- La vérification c'est construire le système correctement (*building the system right* [11]) par rapport aux spécifications. C'est essayer de détecter les erreurs de programmation.
- La validation c'est construire le bon système (*building the right system* [11]). Cela consiste à vérifier que le système construit satisfait les besoins de l'utilisateur

Nous insistons particulièrement sur une qualité de la vérification – héritée du monde des bases de données – à savoir celle de l'« Intégrité ». Elle consiste à maintenir la validité du système évoluant, et ce tout au long de sa vie. Une définition de l'Intégrité donnée par Eaglestone et Ridley cités dans [12] est la suivante :

- L'intégrité, c'est conserver (maintenir) le système de manière à ce qu'il reste correct (*keeping (maintaining) the product right*)[2].

Bien sûr le but, tant dans l'ingénierie du logiciel que dans celui de la connaissance, est de construire le bon système correctement et de maintenir celui-ci.

[2] Le terme d'intégrité, lorsqu'il est considéré en sens large, c'est-à-dire incluant la maintenance, est intéressant pour les SBC. Nous reviendrons plus formellement dans le chapitre consacré aux réponses des bases de données (chapitre 3).

2.2 Nécessité de la vérification et de la validation

Malgré l'essor des systèmes à base de connaissances (SBC), ceux-ci sont toujours difficilement acceptés par le monde de l'industrie. En effet, leur utilisation requiert de l'intuition et de l'expérience dans le domaine. L'acquisition fastidieuse d'informations constitue également un frein à un certain nombre d'applications potentielles. Néanmoins cette technologie est prometteuse et est adoptée par un nombre croissant d'organisation. Bon nombres de recherches sont donc effectuées dans le domaine pour combler les lacunes existantes.

Lors du développement de tels systèmes, ce sont les raisonnements et les connaissances des experts que l'on tente de modéliser. Une des caractéristiques souvent observée des domaines dans lesquels les SBC sont employés, est que les problèmes liés à ces domaines sont mal définis. Des solutions logicielles conventionnelles[3] sont dès lors mal appropriées, tandis que les SBC, bien que ne fournissant pas spécialement la solution optimale - voire ne fournissant pas de solution du tout - sont jugés satisfaisants. Ces systèmes n'ont pas de connaissances meilleures que celles des experts humains qui ont aidés à les construire, mais cela est souvent considéré comme suffisant [13].
De ce fait, la connaissance contenue dans les SBC n'est pas bien définie. Les méthodes pour valider les logiciels conventionnels ne sont donc plus d'application. Pour répondre à ce besoin, beaucoup de méthodologies et d'outils ont été développés pour assister la construction de SBC. Malgré toutes ces recherches, il n'existe toujours pas de modèle clairement établi ni de méthodologie mature, ni surtout de standards de représentation de la connaissance largement acceptés. Au contraire, on retrouve une multitude d'outils de développement différents, la plupart du temps développé pour un SBC bien particulier.

Pourtant, les chercheurs et praticiens du monde des applications basées sur l'intelligence artificielle sont généralement d'accord sur le fait que la phase de vérification et validation

---

[3] On entend par « systèmes conventionnels » ceux dans lesquels la connaissance contenue est suffisamment complète et précise pour assurer l'absence d'erreur. On considérera donc les « systèmes non-conventionnels » comme ceux dans lesquels il existe une ou plusieurs caractéristiques tels que l'incertitude, l'imprécision, l'incomplétude ou l'incohérence, et qui finissent par nécessiter des méthodes spécifiques de représentation conceptuelle et formelle, qui sont la plupart du temps des méthodes heuristiques. (Cardeñosa, J.,Escorial, D.,(1999) « KBS First Prototype V&V Process Plan as a Way to Produce Reliable Exigences »)

(V&V) est une des plus cruciale dans la construction d'un SBC. Il devient donc important d'être capable de vérifier les différents types d'erreurs de la connaissance et l'exactitude du raisonnement du système.

2.3 Typologie des erreurs et anomalies concernées par la V&V

Une liste des principaux types d'erreurs que l'on peut rencontrer dans un SBC est reprise dans le Tableau 2.1 , inspiré de [13], [14] et [15].

Tableau 2.1 *Types d'erreurs fréquents dans les SBC*

Nom	Définition
Circularité, cycles	Se dit lorsqu'il existe un cycle dans les connaissances. Exemple : A → B ; B → C ; C → A
Complétude	Se dit lorsqu'un SBC ne produit pas une conclusion pour tous les inputs.
Conflit	Se dit quand la base de connaissances contient des informations contradictoires. De par cela, le système pourra avoir un comportement erroné. Il faut toutefois noter que des avis contradictoires d'experts sont parfois enrichissants, et qu'il peut donc être intéressant de conserver ces formes de conflits.
Cohérence	Se dit lorsqu'un SBC ne produit pas un ensemble cohérent de conclusion pour tous les inputs, c'est-à-dire, que pour chaque ensemble d'inputs possible, toutes les conclusions ne sont pas vraies en même temps.
Déficience	Se produit lorsqu'il y a des entrées valides vers une base de connaissances qui ne contient aucune règle relative à cette connaissance.
Exactitude, fiabilité	Se dit lorsque la connaissance détenue par les experts est mal représentée.

Redondance	Se dit quand la base de connaissance contient la même connaissance représentée à plusieurs endroits ou qu'il existe des informations qui ne jouent pas sur le comportement de la résolution du problème par le système (mais qui peuvent être vraies).

L'objectif premier va être d'obtenir un système qui est d'une part valide, c'est-à-dire qui résout les problèmes correctement comme l'aurait fait un expert humain. Et d'autre part, qui est complet et qui résout tous les problèmes que peut rencontrer l'utilisateur. La nécessité d'une phase de V&V est donc assez évidente.

2.4 Influence de la V&V dans le succès des systèmes à base de connaissances

Les techniques de V&V permettent de mesurer la qualité du contenu d'une base de connaissances. Ainsi, la vérification nous dit si une base de connaissances est endommagée, tandis que la validation nous dira plutôt si la connaissance (fournie par les experts humains) est fidèlement représentée. Nous constatons que la vérification est essentiellement un test objectif, tandis que la validation à plus un caractère subjectif, dépendant du contexte, où l'on doit comparer des objets formellement représentés à des expressions informelles.

Preece [13] a identifié les trois facteurs de succès d'un SBC :

- Avons-nous construit le bon système : avons-nous satisfait les exigences des utilisateurs ?
- Savons-nous le garder ainsi : le système est-il suffisamment maintenable pour anticiper de futurs changements ?
- Savons-nous le refaire : est-ce qu'il est possible de refaire ce processus pour assurer le succès de futurs projets ?

Le processus de V&V tel que défini à la section 2.1 répond à la première de ces trois questions. Mais, comme le précise Preece, le tout n'est pas de construire un système correct, encore faut-il que ce système soit maintenable et maintenu. Pour ce faire, le concept d'intégrité au sens large, également défini en 2.1, se veut une réponse à la deuxième question. Il faut toutefois noter que le processus de V&V en tant que tel répond partiellement à ces deux dernières questions dans le sens où il permet une maintenabilité plus aisée et dans le sens où

un processus répétable de V&V est un pré-requis pour assurer le succès dans l'ingénierie de la connaissance.

Le dernier point, quant à lui, reflète l'aptitude de l'ingénieur de la connaissance, ainsi que le point de vue, assez récemment apparu, que la qualité des logiciels est principalement déterminée par la qualité du processus de développement :

> « *C'est un fait largement accepté que la qualité d'un produit logiciel est largement déterminé par la qualité du processus utilisé pour développer et maintenir celui-ci* » [16]

Ceci est également valable pour les SBC. Une attention particulière sur la qualité doit donc être appliquée tout au long du processus de développement et du processus de maintenance.

Nous pouvons constater que le processus de V&V est indispensable dans le cycle de vie d'un SBC. Nous allons maintenant montrer l'importance de ce processus tout d'abord dans le cycle de développement même du système, ensuite dans la phase de maintenance. Ces deux points seront les sujets de la prochaine section.

2.5 Importance de la vérification et de la validation dans le cycle de vie d'un SBC

Jusqu'à présent nous avons principalement attiré l'attention sur l'importance de la vérification et de la validation comme technique de mesure de la qualité. Pourtant, d'autres aspects sont à prendre en compte, comme le précise Preece [17] :

> « *Tout comme dans l'ingénierie du logiciel, l'ingénierie de la connaissance peut être vue comme une combinaison de méthodes et de processus de mesures : les méthodes utilisées dans les spécifications des exigences, dans l'acquisition et la modélisation de la connaissance, dans l'architecture du système, et dans l'implémentation de celui-ci résultent en la production d'une série d'artefacts.* »

Chacun de ces artefacts est sujet à une certaine forme de mesure et les techniques de V&V fournissent les moyens d'obtenir ces mesures. Pour rappel, les artefacts suivants sont nécessaires dans le processus de développement d'un logiciel conventionnel :

- Spécifications des exigences des utilisateurs
- Modèle conceptuel
- Modèle architectural
- Système implémenté

Ces étapes doivent être quelques peu adaptées pour le développement d'un SBC. En effet, une des étapes caractérisant le développement de tels systèmes est la modélisation de la connaissance. Combiné avec le processus de développement d'un SBC proposé par Mattei [18], la Figure 2.1 représente le cycle de vie d'un SBC.

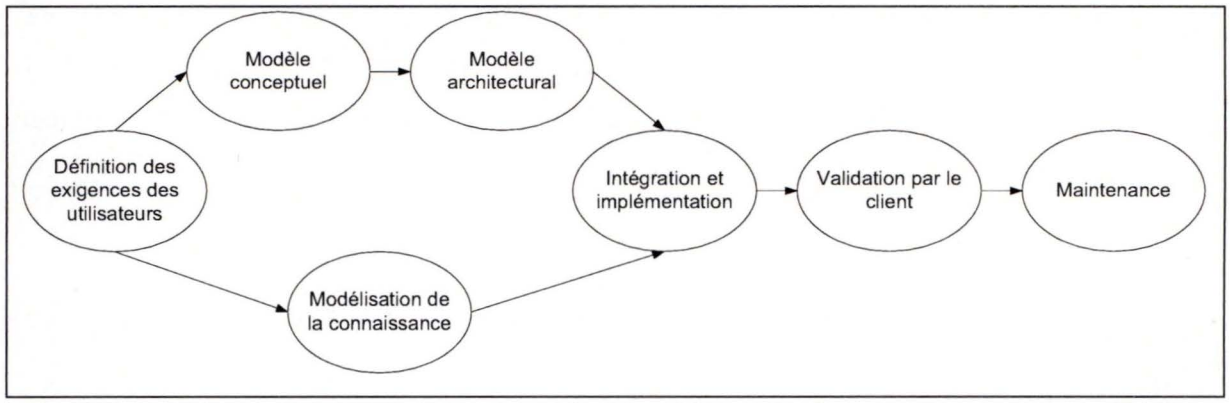


Figure 2.1 *Activités principales dans le développement d'un système à base de connaissances*

Une détection des erreurs, dès le début du processus de développement, est devenue plus importante, surtout parce que la technologie de la connaissance est de plus en plus utilisée dans des environnements dits « critiques »[4].

Dès lors, une idée apparue depuis longtemps est que les techniques de V&V peuvent s'appliquer directement sur les exigences des utilisateurs. Seulement, comme nous l'avons déjà fait remarquer, une des caractéristiques des domaines des SBC est que les problèmes liés à ceux-ci sont mal définis. Il en résulte que les exigences des utilisateurs sont mal définies. Des techniques vont de ce fait être développées de manière à les représenter par des spécifications formelles. A partir de celles-ci, il pourra être démontré que le système implémenté rencontre ces différentes exigences.

---

[4] Un système critique est un système dans lequel un défaut de fiabilité peut engendrer des pertes importantes.

Malheureusement, comme Preece [13] le fait remarquer, deux barrières vont s'opposer à ceci :

- Même s'il est possible de formaliser ces exigences, cela peut être trop difficile de créer ces spécifications endéans le temps et le budget imparti au projet.
- Il y a beaucoup de types d'exigences qui ne sont pas sujets à des spécifications formelles (par exemple : « l'utilisabilité » d'une interface graphique).

Nous pouvons donc constater qu'une vérification effectuée uniquement par le biais des spécifications ne sera pas suffisante. Le processus de V&V doit donc utiliser d'autres techniques sur les phases de développement suivantes. Pour arriver à un SBC de haute qualité, le processus de V&V est devenu nécessaire dans toutes les phases du cycle de vie de l'ingénierie de la connaissance :

- Dans la phase de modélisation de la connaissance, il faut valider les connaissances avec un expert ;
- Dans la phase d'implémentation, il faut vérifier l'exactitude logique des connaissances implémentées ;
- Dans la phase de validation par le client, il faut utiliser des cas de test (« *tests cases* ») afin de valider la base de connaissances.

Des outils de V&V peuvent être utilisés dans chaque phase du cycle. Par exemple :

- Du datamining peut être utilisé pour assister le processus d'extraction de la connaissance par la génération automatique de règle. Cette technique de datamining utilise la connaissance des bases de données pour valider la connaissance de l'expert en cherchant des preuves basées sur des données disponible [19].
- Des outils de vérification automatique de la base de connaissance pour les connaissances inconsistantes, redondantes ou incomplètes, et ce pendant le développement du SBC.
- Une génération automatique des cas de test pour une base de connaissance.

Ces outils vont augmenter la qualité et réduire le temps pour la mise en place d'un SBC. Néanmoins, ces outils sont la plupart du temps spécifiques à un SBC particulier.

2.5.1 Techniques de V&V

Bon nombre de techniques de V&V ont été développées dans le monde du développement logiciel. Certaines ont été adaptées aux SBC. Nous en exposons brièvement les plus habituelles ci-dessous, en précisant d'une part leur rôle dans les différentes étapes du développement d'un SBC, et d'autre part leur rôle pour une gestion de la connaissance[5].

2.5.1.1 Inspection

D'après une étude effectuée par O'Leary cité dans [20] sur des développeurs de SBC pour des applications commerciales, c'est la technique la plus habituellement employée. Mais c'est aussi la moins fiable. En effet, il s'agit la plupart du temps d'une simple relecture par un expert d'un ou plusieurs artefacts.
Puisque les langages formels utilisés dans les phases de modélisation architecturale et d'implémentation sont souvent incompris des experts, cette technique est plus adéquate pour l'étape de conception du modèle.

Pour la gestion de la connaissance, cette technique est très appropriée dans le sens où il y a une véritable nécessité de relecture par les experts de la connaissance acquise par les ingénieurs de la connaissance (cf. Figure 1.1). C'est aussi la forme minimale de vérification qui doit être appliquée dans tout projet de gestion de la connaissance.

2.5.1.2 Vérification statique

Cette technique consiste à vérifier que la base de connaissances ne comporte pas d'anomalies logiques. Elle est fortement utilisée dans les SBC basés sur un système de règles, et de nombreux outils existent à cette fin. Les types d'erreurs les plus détectées par ce genre d'outils sont la redondance et le conflit. Notons que ces anomalies peuvent apparaître à tous les niveaux de la conception, mis à part dans les spécifications des exigences.

Une application possible de ce type de vérification au niveau de la gestion de la connaissance est de pouvoir vérifier les différents avis contradictoires des experts. Cependant, comme nous

---

[5] L'expression « gestion de la connaissance » désigne les opérations de création, modification, et suppression de connaissances. En d'autres termes, il s'agit de maintenance de la connaissance du système. Ces opérations se retrouveront donc principalement dans les phases de modélisation de la connaissance et de maintenance.

l'avons déjà fait remarquer, il peut parfois être intéressant de garder ces différents avis pour une meilleure compréhension de la connaissance ou en vue d'une réingénierie.

2.5.1.3 Vérification formelle

C'est une vérification logique plus minutieuse que la technique de vérification statique. Elle peut être appliquée là où les spécifications des exigences des utilisateurs peuvent être décrites de manière formelle. On peut alors vérifier que les artefacts formels correspondent aux exigences spécifiées. En d'autres mots, cela permet de vérifier la conformité d'un logiciel (ou d'une base de connaissances) par rapport à ses spécifications formelles.

En ce qui concerne la gestion de la connaissance, vu la difficulté de mise en œuvre des langages formels ou des techniques de preuves logiques, celle-ci ne se justifie seulement que là où la connaissance est appliquée dans des domaines critiques ou coûteux.

2.5.1.4 Vérification par références croisées

Lorsqu'il existe des descriptions du SBC à différents niveaux, il peut être fort intéressant de s'assurer qu'il y a correspondance entre ces différentes couches, et ce de manière à assurer la cohérence et un état complet du système. Par exemple, les concepts spécifiés au niveau conceptuel devront se retrouver sous forme d'entité concrète au niveau de l'architecture, et sous forme de structures de données dans le système implémenté. De par cela, l'application de cette technique se retrouvera presque exclusivement entre :

- Le modèle conceptuel et le design du modèle
- Le design du modèle et le système implémenté

Du point de vue de la gestion de la connaissance, s'il existe une documentation claire et précise des différents niveaux de conceptualisation, alors cela permet à l'ingénieur de la connaissance (si c'est celui-ci qui effectue la maintenance) de naviguer entre les différents niveaux pour une meilleure compréhension de la connaissance et une mise à jour de celle-ci si nécessaire. De ce fait, cette technique assure une certaine cohérence du SBC.

2.5.1.5 Tests empiriques

La définition du test établie par l'IEEE- Standard Glossary of Software Engineering terminology IEEE-STD729 (1983) est la suivante :

« Le test est l'exécution ou l'évaluation d'un système ou d'un composant par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus ».

Pour cela il faut élaborer différents cas de tests et analyser ceux-ci.
On retrouve dans la littérature deux grandes classes de tests : les tests fonctionnels et les tests structurels.
Dans le cas des tests sur les fonctionnalités du système, dits de « boîte noire », aucune attention n'est prêtée à la manière dont celui-ci est implémenté. Les données de tests sont souvent basées sur les documents de spécification. Le but des tests fonctionnels est d'exercer chaque aspect du comportement du système par l'intermédiaire d'un sous-ensemble de vecteurs d'entrées possibles. Il est, en général, impossible de tester un programme avec tous les vecteurs d'entrées théoriquement valables pour cause d'explosion combinatoire. Le succès d'une telle technique est donc directement relatif à l'élaboration d'ensembles de tests suffisamment représentatifs des différentes possibilités du système.
Les tests structurels, dits de « boîte blanche », comme leur nom l'indique, testent la structure du système. Les données de tests sont donc basées sur l'implémentation du système. Encore une fois, on ne peut pas tester tous les chemins d'exécution d'un logiciel, d'où l'importance d'ensembles de tests représentatifs.

Au niveau de la gestion de la connaissance, les tests se résument à questionner systématiquement le SBC dans le but d'évaluer l'acceptabilité des réponses en terme de complétude et d'exactitude. Nous retrouvons ici le soucis cité dans [21] que « *les systèmes à base de connaissances doivent produire une réponse, une solution, ou un comportement qui est équivalent à ceux produit par les meilleurs experts humain.* »

2.5.2 Applications des techniques dans le processus de développement

Nous croyons cependant que toutes ces techniques ne sont pas applicables à toutes les étapes du processus de développement. Le Tableau 2.2 extrait de [13] reprend les techniques énoncées ci-dessus et les associe aux artefacts du processus de développement d'un SBC auxquels elles peuvent s'appliquer.

Tableau 2.2 *Applications potentielles des différentes techniques de V&V aux artefacts.*

Artefact	V&V techniques
Modélisation de la connaissance	• Inspection • Vérification par références croisées (s'il existe plusieurs niveaux de conceptualisation de la connaissance).
Modèle conceptuel	• Inspection • Vérification statique (si formalisé) • Vérification par références croisées (par rapport au design du modèle)
Design du modèle	• Inspection • Vérification statique • Preuve formelle • Vérification par références croisées (par rapport au modèle conceptuel et au système implémenté)
Système implémenté	• Inspection • Vérification statique • Tests • Vérification par références croisées (par rapport au modèle architectural)
Maintenance	• Inspection • Vérification statique • Tests

A la lecture de ce tableau, on peut se demander quelle technique de V&V fonctionne le mieux et avec quelle méthode de création des artefacts. Cette question reconnaît le fait que toutes ces différentes techniques ne peuvent pas être appliquées à n'importe quelle méthode. Par exemple, la vérification statique pour vérifier les anomalies logiques ne peut être appliquée à un système implémenté que si celui-ci a été crée dans un langage de programmation qui permet de définir ces « anomalies logiques ». De même, la technique de preuve formelle ne peut être appliquée au design du modèle que s'il existe une théorie appropriée pour ce langage modélisé.

2.6 La maintenance d'un système à base de connaissances par un expert du domaine

2.6.1 Introduction

Pendant la maintenance, suite au changement d'une seule connaissance, il est possible d'introduire des contradictions, des redondances ou des incomplétudes dans une chaîne complexe de connaissances. La maintenabilité a dès lors une place toute particulière dans la gestion de la connaissance, considérant que les bases de connaissances d'une organisation vont nécessairement évoluer.

Avoir recours directement à un expert du domaine pour maintenir la connaissance d'un SBC constitue la solution idéale. Le souhait de l'ingénierie de la connaissance est de vouloir supprimer l'ingénieur de la connaissance du processus du cycle de maintenance et de fournir des outils permettant à l'expert du domaine de maintenir par lui-même le système et de l'ajuster aux changements de la connaissance.

2.6.2 Motivation

Comme nous l'avons déjà mentionné, les systèmes basés sur la connaissance connaissent un succès grandissant auprès des entreprises, et ce malgré les diverses barrières à leur utilisation.

Parmi ces barrières, on retrouve le fait que l'utilisation de SBC requiert de l'intuition et de l'expérience dans le domaine, l'acquisition de données en est dès lors rendue difficile. Une autre des principales barrières à la mise en place d'un tel système sont les coûts de maintenance. Ceux-ci représentent souvent près de 80% des coûts totaux quand on considère l'entièreté de la vie d'un projet logiciel. Une réponse à cette problématique, apparue au fur et à mesure de l'évolution des SBC, est, à nouveau, d'éliminer l'ingénieur de la connaissance du processus.

Le processus d'ingénierie de la connaissance implique traditionnellement au minimum les trois parties suivantes :

- L'utilisateur final du système
- L'expert du système qui fournit l'expertise de la base du système
- L'ingénieur de la connaissance qui acquiert la connaissance de l'expert du domaine et qui l'utilise pour construire le système

Tel que développé à la section 1.1.3.(acteurs des systèmes experts), c'est à l'ingénieur de la connaissance qu'incombe la tâche la plus difficile, surtout quand le domaine de connaissance est lui-même mal défini, voire mal compris. Une des principales difficultés rencontrées par cet acteur est de réussir à comprendre et à faire exprimer ce que les experts détiennent comme connaissance. Une autre difficulté est de comprendre ce que les utilisateurs finaux désirent (ceux-ci sont souvent incapables de définir clairement leurs attentes). Ainsi, l'ingénieur de la connaissance doit donc implémenter un système fondé sur une connaissance incertaine et ce de manière à respecter les exigences souvent mal définies des utilisateurs.

Dans la situation traditionnelle, tout changement requis dans la base de connaissances doit être communiqué par l'expert du domaine à l'équipe de maintenance, formée d'ingénieurs de la connaissance, qui vont eux-mêmes implémenter les changements. Ceux-ci devront par la suite être vérifiés et validés par l'expert du domaine (qui pourra suggérer des modifications). Ce sera seulement à la fin de ce processus itératif que la nouvelle version du système pourra être fournie aux utilisateurs finaux. On voit donc que le processus de maintenance correspond au processus de construction du système lui-même (Figure 1.1).

Le souhait de l'ingénierie de la connaissance se traduit donc tout naturellement à vouloir supprimer cet intermédiaire qu'est l'ingénieur de la connaissance du processus du cycle de maintenance et ce en fournissant aux experts du domaine des outils adéquats leur permettant d'implémenter eux-mêmes directement les changements voulus. Ces outils assureront un gain de temps considérable (il n'y aura plus de processus répétitifs) et une qualité de la solution (plus de mauvaises communications, compréhensions entre experts et ingénieurs).

2.7 Définition de la problématique

Maintenant que les principaux concepts utiles ont été redéfinis, nous pouvons délimiter nos perspectives de recherche.

Nous avons vu que le monde de l'industrie est intéressé par l'utilisation des systèmes experts ou des SBC car ils apportent des solutions que n'offrent pas les logiciels conventionnels. Pourtant, cet attrait n'est pas aussi important qu'il pourrait l'être, du fait des nombreuses difficultés qui subsistent encore. Nous pouvons citer la difficulté d'acquérir et de transférer la connaissance, le fait que l'utilisation des SBC requiert de l'intuition et de l'expérience dans le domaine.

L'une des voies de recherche qui nous semble importante, est de supprimer l'ingénieur de la connaissance du processus de développement d'un SBC. En effet, pour l'instant, c'est ce dernier qui est habilité à développer, ou à maintenir le système après avoir extrait la connaissance des experts du domaine (cf. Figure 1.1). Or, cette étape peut entraîner des ambiguïtés. Si l'on permet aux experts de développer/maintenir directement le système, ils seront plus à même de modéliser la connaissance. Toutefois, pour atteindre cet objectif, il est nécessaire de simplifier l'utilisation des SBC.

Cette problématique étant particulièrement étendue, nous allons nous focaliser sur l'utilisation de concepts bien définis issus du monde des bases de données pour permettre la maintenance d'un SBC existant. Nous nous reposerons également sur l'introduction d'une interface homme-machine intelligente. Celle-ci doit non seulement permettre l'introduction de nouvelles connaissances, mais également vérifier de manière automatique la cohérence de la base de connaissances. Il en va de même pour toutes les opérations possibles (modification, suppression).

Chapitre 3 : La réponse des Bases de Données

Comme nous venons de le souligner dans le chapitre précédent, les systèmes experts et plus précisément les systèmes à base de connaissances, souffrent d'une faiblesse importante au niveau de la maintenabilité des connaissances. Il n'existe en effet aucun mécanisme permettant de les modifier, tout en garantissant de façon dynamique (c'est-à-dire au moment même de l'introduction de la modification) leur intégrité, ainsi que leur cohérence avec le reste des connaissances.

L'univers des bases de données quant à lui, regorge de tels mécanismes. Ils constituent un sous-ensemble des fonctionnalités nécessaires de tout *Système de Gestion de Bases de Données* (SGBD).

Cette section est consacrée à une présentation de ces mécanismes généralement adoptés dans le monde des bases de données. Ces principes seront adaptés et utilisés par la suite dans la réalisation de notre application à CosmicXpert, dans le chapitre 7.

Les concepts présentés ici sont tirés de [22] et de [23].

3.1 Les systèmes de Gestion de Données

Une base de données est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondances possible. Ces données doivent pouvoir être utilisées par des programmes ainsi que par des utilisateurs différents. Ainsi, la notion de base de données est généralement couplée à celle de réseau, afin de pouvoir mettre en commun ces informations, d'où le nom de base. On parle généralement de système d'information pour désigner toute la structure regroupant les moyens mis en place pour pouvoir partager des données.

Une base de données permet de mettre des données à la disposition d'utilisateurs pour une consultation, une saisie ou bien une mise à jour, tout en s'assurant des droits accordés à ces derniers.

La gestion de la base de données se fait grâce à un système appelé SGBD (*Système de Gestion de Bases de Données*) ou en anglais DBMS (*DataBase Management System*). Le SGBD est un ensemble de services (applications logicielles) permettant de gérer les bases de données, c'est-à-dire :

- garantir la qualité des données enregistrées,
- leur cohérence,
- les protéger en cas d'incidents,
- permettre à plusieurs utilisateurs d'y accéder simultanément,
- tout en contrôlant l'accès aux données confidentielles,
- offrir de bonnes performances d'accès à toutes les applications.

En résumé, nous pouvons définir un SGBD comme :

> *« un système logiciel assurant, entre autres, les fonctions de définition des structures de données, de leur évolution, de gestion des données, d'accès aux données, de contrôle de l'intégrité, du contrôle d'accès, de la protection contre les incidents, de la régulation de la concurrence. »*[23]

3.2 Les transactions

Le mécanisme de transaction, comme le montre cette section, est particulièrement intéressant pour garantir l'intégrité d'une base de données. Ce mécanisme sera à la base de notre travail, dans le chapitre 7, dans l'application aux bases de connaissances, et plus particulièrement à CosmicXpert.

3.2.1 Définition et objectif des transactions

Le SGBD permet entre autres la gestion d'une fonctionnalité particulièrement intéressante dans le cadre de la gestion des incidents : le principe de transaction. Une transaction peut être définie comme *« la section d'un programme constituant une unité logique de traitement du point de vue des accès à la base de données »* [22].

Une transaction est une unité de traitement cohérente et protégée. Elle vérifie, relativement à la base de données, les quatre propriétés A.C.I.D, décrites dans [22] :

- Atomicité (*Atomicity*) : l'ensemble des instructions de la section sont exécutées complètement (en cas de succès), ou pas du tout (en cas d'incident), mais jamais partiellement ;
- Cohérence (*Consistency*) : si la base de données est cohérente avant la transaction, elle le sera après la sortie de celle-ci (respect des contraintes d'intégrité) ;
- Indépendance (*Isolation*) : les interactions entre la transaction et la base de données s'effectuent comme si la transaction était seule à travailler sur la base de données (pas d'interférences nuisibles avec les autres programmes) ;
- Durabilité, permanence (*Durability*) : si la transaction s'est terminée correctement, les modifications faites dans la base de données sont garanties permanentes (sauf modifications via d'autres transactions), quels que soient les incidents ultérieurs.

Certaines instructions particulières vérifient les propriétés A.C.I.D, il s'agit des *instructions* ou *requêtes primitives*. Le SGBD garantit (contractuellement), dans ce cas, que les propriétés sont satisfaites.

Cependant, la granularité des primitives peut être trop fine pour certaines opérations sur les données. En effet, comme le montre la Figure 3.1, on peut observer que si un incident survient avant le retrait dans le compte source, il n'y aura aucune répercussion sur l'intégrité de la base de données. De même, si un incident survient après la mise à jour du compte destination. Par contre, si un incident se produit entre le retrait sur le compte source et l'ajout dans le compte destination, alors l'intégrité de la base de données ne pourra pas être assuré.

```
                    (Si incident : OK)
        update du compte source : - 1000 €
                (Si incident : incohérence)
        update du compte destination : + 1000 €
                    (Si incident : OK)
```

Figure 3.1 *Le transfert d'une somme d'un compte vers un autre nécessite deux primitives*

Le mécanisme de transaction permet de répondre à ce problème par le fait que la transaction constitue une *super primitive* construite par le programmeur. Une transaction est un ensemble de primitives exécutée comme étant une seule et même primitive, elle constitue de ce fait une super primitive.
Le résultat de la mise à jour des deux comptes avec le mécanisme de transaction est présenté à la Figure 3.2.

```
                    (Si incident : OK)
        primitive(
                update du compte source : - 1000 €
                update du compte destination : + 1000 €
                )
                    (Si incident : OK)
```

Figure 3.2 *Solution avec utilisation d'une transaction*

Une transaction est effectuée dans l'univers des bases de données, à l'aide de trois primitives offertes par le SGBD :

- Ouverture d'une transaction : `begin-transaction`
- Clôture d'une transaction avec confirmation : `commit-transaction`
- Clôture d'une transaction avec annulation : `abort-transaction`

Si un programmeur désire que les propriétés A.C.I.D. soient vérifiées pour une séquence d'instructions, il doit en faire une transaction.

3.2.2 Structure des transactions

De manière générale, une transaction possède la structure fonctionnelle présentée à la Figure 3.3. C'est-à-dire, pour chaque unité de données à traiter :

- on saisit les données, ce qui nécessite en général un dialogue avec l'utilisateur,
- on les valide, via notamment des accès à la base de données,
- on modifie la base de données selon les données acquises et validées.

```
préparation
begin-transaction
        acquisition des données
        validation des données
        modification des données
        si erreur abort-transaction
commit-transaction
suite du traitement
```

Figure 3.3 *Schéma général d'une transaction*

Il est à noter qu'à cette structure générale peut correspondre plusieurs organisations techniques fonctionnellement équivalentes, mais de comportements différents selon l'environnement (acquisition et/ou validation des données à l'intérieur ou à l'extérieur de la transaction, contrôle ou non de l'état des données dans la transaction, ...).

3.2.3 Transactions hiérarchiques

Tel que précisé en 3.2.1, une transaction constitue une super primitive. Lors de l'exécution d'un programme, les transactions se déroulent séquentiellement mais jamais en parallèle. Il n'est donc pas possible de définir une transaction comme étant formée de sous-transactions plus élémentaires. Une structure hiérarchique n'étant pas possible, on peut affirmer que la structure d'une transaction ne correspond pas à la structure hiérarchique d'un programme. Cette faiblesse d'architecture logicielle fait apparaître plusieurs inconvénients, comme l'impossibilité de réutilisation ou la dépendance des modules.
Illustrons ces propos par un exemple. Soit la fonction d'ouverture d'un compte. Il existe tout d'abord une transaction qui crée le compte.

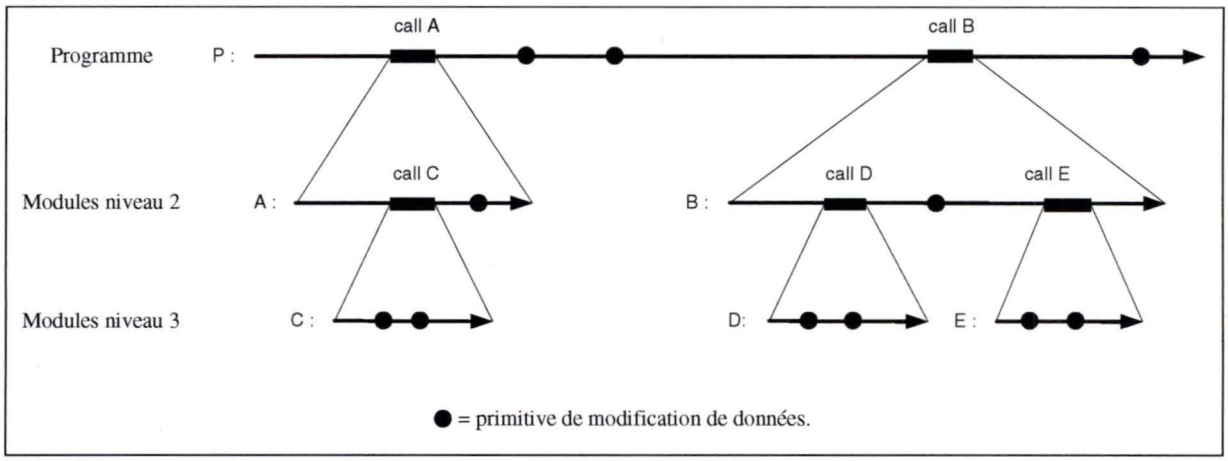


Figure 3.4 *Structure d'un programme hiérarchisé*

Cas 1 : Cette première transaction peut ensuite appeler toute une série de *procédures* pour initialiser le compte, etc. Si l'une de ces procédures échoue, il faudra annuler toute l'exécution et recommencer depuis le début.

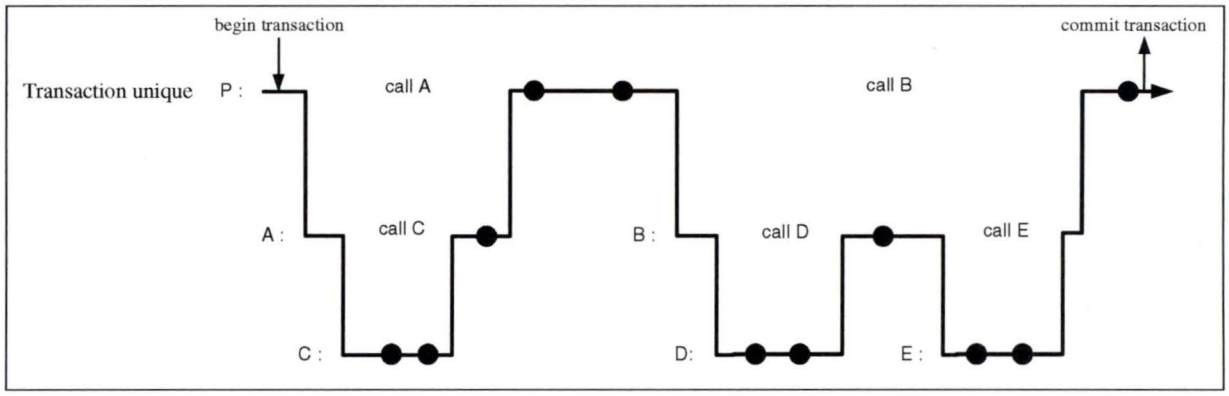


Figure 3.5 *Cas 1 : Structure transactionnelle plate*

Cas 2 : Cette première transaction peut ensuite appeler d'autres *transactions* pour initialiser le compte, etc. Si une des transactions échoue, il n'est pas nécessaire de recommencer toutes les transactions, seule celle qui a échoué devra être réexécutée.

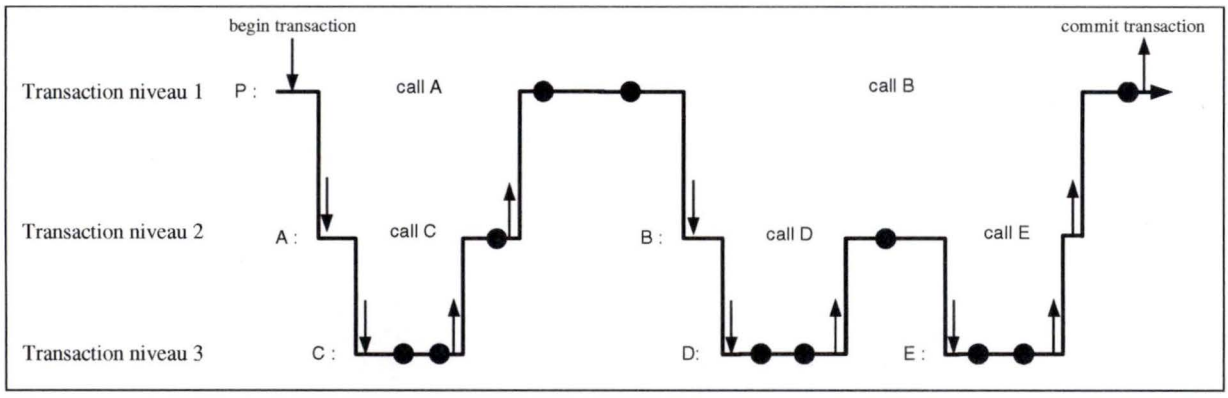


Figure 3.6 *Cas 2 : Structure transactionnelle hiérarchisée*

3.2.4 Transactions et intégrité

Nous avons vu qu'une transaction doit satisfaire aux quatre propriétés A.C.I.D. et donc, plus particulièrement, garantir la cohérence de la base de données. Plus précisément, si la base de données est cohérente à l'entrée de la transaction, alors, au sortir de la transaction,

- soit la base est encore dans cet état (abort, incident),
- soit elle se trouve dans un nouvel état (commit) qui respecte les contraintes d'intégrité, comme les *primary key*, *unique*, *foreign key*, *not null*[6], ...

Qu'en est-il des contraintes qui ne sont satisfaites que suite à une séquence de plusieurs primitives ? Par exemple, dans le cas d'un transfert d'un compte vers un autre (contrainte : la somme des comptes doit être invariante). La contrainte n'est pas satisfaite après la modification du premier compte. La base de données se trouve donc dans un état incohérent après cette première étape. Une solution à ce problème, proposée par les SGBD, est de postposer la vérification des contraintes, lors d'une transaction.

3.3 Protection contre les incidents

Jusqu'à présent, nous avons souligné les différents rôles joués par le SGBD, afin de vérifier les propriétés A.C.I.D. de la base de données. Dans cette nouvelle section, nous allons tout particulièrement insister sur la garantie des propriétés d'Atomicité et de Durabilité. En effet,

---

[6] Les termes utilisés ici respectent la syntaxe SQL et expriment un ensemble de contraintes d'intégrité. Pour plus d'informations, lire [22] J.-L. Hainaut, "DB-MAIN, Ingénierie des bases de données, matières approfondies." Namur, 2000, pp. 3.2-5.26.

ces deux propriétés d'une transaction relèvent d'un autre rôle du SGBD : la protection contre les incidents.

3.3.1 Principes

Avant toute chose, il est nécessaire de définir un incident. Il s'agit de façon générale, « *d'un événement fortuit ou intentionnel pouvant affecter l'intégrité de la base de données* ». [22]
Tout SGBD doit être capable de réparer des données corrompues suite à un incident.
La propriété d'Atomicité précise qu'en cas d'incident au sein de la transaction, la base de données doit être remise dans l'état du début de la transaction. Si des modifications ont déjà été réalisées, elles doivent être défaites (*rollback*).
La propriété de Durabilité, quant à elle, demande qu'en cas d'incident après la clôture de la transaction, les données, même corrompues, doivent être remises au moins dans l'état où elles étaient au sortir de la transaction.
Cette double protection est assurée de façon générale dans l'univers des bases de données par la disponibilité de deux fichiers annexes : la sauvegarde (*backup*) et le journal (*log*).

3.3.2 La sauvegarde

La sauvegarde, ou *backup*, est une copie partielle ou totale de la base de données à un instant de référence.
La copie complète à l'avantage de proposer une version prête à l'emploi. Par contre, cette copie nécessite beaucoup de temps. Elle est généralement compactée et stockée sur un support sûr.
La copie partielle ou incrémentale ne reprend que les parties ayant été modifiées depuis la dernière sauvegarde. Elle suppose l'existence d'une copie complète initiale. Elle nécessite moins de temps mais ne propose pas de version complète de la base de données. Celle-ci doit être reconstituée par application des incréments successifs à la copie complète initiale.

3.3.3 Le journal

Le journal, ou *log*, est un fichier dans lequel sont consignées toutes les modifications effectuées dans la base de données, ainsi que l'historique des transactions. Les modifications sont représentées par une copie des données avant et après l'opération.

Il existe plusieurs types de fichier journal :

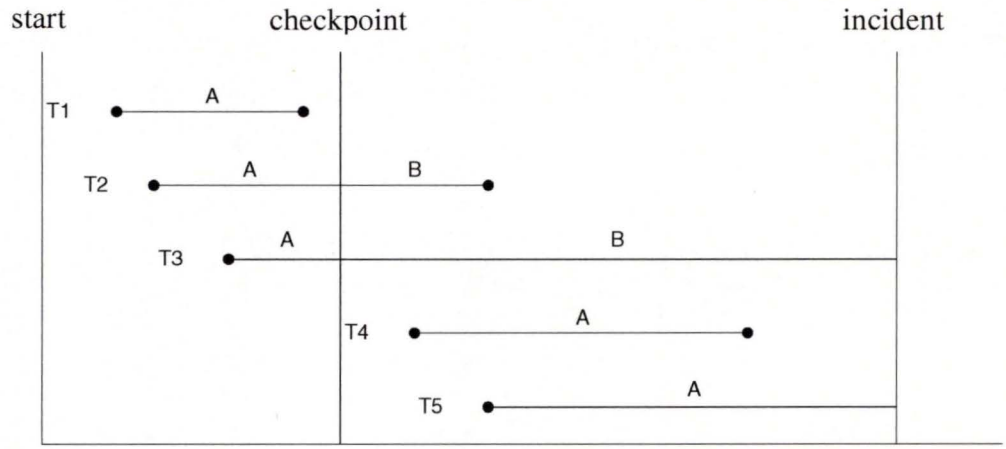
- Journal des images *avant* : contient la copie des pages avant leur modification. Il permet de défaire des modifications qui ne se sont pas terminées correctement.
- Journal des images *après* : contient la copie des pages après leur modification. Il permet de refaire des modifications qui ne se sont pas terminées correctement.
- Journal *mixte* : contient les copies des pages avant et après leur modification, de façon à assurer tous les types de reprises.

3.3.4 Reprise suite à un incident

La reprise *à froid* est de mise lorsque la base de données a subi un incident grave, la détériorant gravement, voire la rendant inutilisable. Elle se caractérise par deux étapes :

- récupérer la dernière sauvegarde,
- réappliquer les images *après* enregistrées dans le journal depuis la prise de cette sauvegarde. Seules les images correspondant à des transactions clôturées sont réappliquées. Le journal est parcouru à l'envers de façon à ne réappliquer que la dernière version de chaque page.

Si la base de données est localement détériorée, il n'est pas nécessaire de suivre le schéma précédent, jugé trop lourd. On lui préférera une correction *à chaud*. Les différents scénarios possibles sont repris à la Figure 3.7.



Type de transaction	Etat	Correction
T1	T1 permanente	---
T2	A permanent, B incertain	undo(B), redo(B)
T3	A permanent, B incertain	undo(T3)
T4	T4 incertain	undo(T4), redo(T4)
T5	T5 incomplète	undo(T5)

Figure 3.7 *Schéma temporel d'un incident et reprise à chaud*

3.4 Régulation de la concurrence

Dans la section précédente, nous nous sommes attardés sur les propriétés d'Atomicité et de Durabilité. Celle-ci est consacrée à la propriété d'Isolation qui précise que chaque transaction doit se comporter comme si elle était seule à travailler sur la base de données.

3.4.1 Définition de la problématique

On dit que des transactions sont concurrentes lorsqu'elles accèdent simultanément à la base de données et qu'elles utilisent les mêmes données.

Si parmi les transactions concurrentes, au moins une modifie les données, alors des interférences nuisibles peuvent affecter l'exécution de ces transactions.

Ces interférences sont au nombre de trois :

- Un processus peut modifier des données qu'un autre processus est en train de modifier. On parle dans ce cas de la mise à jour perdue (*lost update*).
- Un processus peut accéder à des données instables, c'est-à-dire, modifiées par une transaction qui n'a pas encore confirmé ces modifications.
- Un processus peut modifier une collection de données dont un autre processus analyse l'état. On parle alors d'incohérence statistique.

3.4.2 Solutions

Plusieurs classes de techniques existent pour éviter les interférences nuisibles.
Les premières consistent à évaluer en temps réel les états des transactions. Ces techniques sont, en pratique, beaucoup trop coûteuses en calcul.
Les secondes imposent des protocoles restrictifs qui évitent les interférences. Ces techniques sont plus économiques mais peuvent limiter le parallélisme. L'exemple type de cette famille est le principe du *verrouillage*.

3.4.3 Technique de verrouillage

La technique de verrouillage est régie par quelques grands principes :

- une transaction peut poser un verrou sur une donnée,
- il existe deux types de verrous : partagés et exclusifs,
- avant de lire une donnée, la transaction doit obtenir un verrou partagé sur cette donnée,
- avant de modifier une donnée, la transaction doit obtenir un verrou exclusif sur cette donnée,
- une transaction peut obtenir un verrou partagé sur une donnée si celle-ci ne possède pas de verrou exclusif,
- une transaction peut obtenir un verrou exclusif sur une donnée si celle-ci ne possède pas de verrous posés par d'autres transactions.

Ces principes ont pour conséquences que si une transaction a obtenu un verrou partagé sur une donnée, une autre transaction peut lire cette donnée, mais pas la modifier. De même, si

une transaction a obtenu un verrou exclusif sur une donnée, une autre transaction ne peut ni lire ni modifier cette donnée.

En pratique, c'est le SGBD qui gère ces demandes de verrous, qui ne sont pas exprimées par des primitives explicites. De même, la demande de libération d'un verrou est implicite, à la fin de l'exécution de la primitive. Il faut toutefois mentionner le problème de l'interblocage qui survient lorsqu'une primitive bloque des données nécessaires à une autre primitive, qui elle-même bloque des données qui sont requise par la première, rendant les deux exécutions impossibles. Ce problème devra lui aussi être résolu par le SGBD.

Chapitre 4 : Présentation de CosmicXpert

Ce chapitre, ainsi que les deux suivants, ont pour objectif de présenter CosmicXpert, le système expert qui est à l'origine de ce travail. Nous aborderons ce système en exposant, dans un premier temps, sa raison d'être ainsi que le premier prototype. Nous poursuivrons ensuite par la présentation du prototype 2 (chapitre 5). Nous terminerons cette présentation par le relevé des faiblesses de ce deuxième prototype et la description de notre problématique (chapitre 6).

CosmicXpert est l'aboutissement de la thèse de doctorat de Jean-Marc Desharnais. Cette thèse a été réalisée dans les laboratoires de l'Université du Québec à Montréal (UQAM). Elle décrit, entre autres, la démarche cognitive du mesureur utilisant la méthode de mesure fonctionnelle COSMIC-FFP (*Common Software Measurement International Consortium – Full Function Points*) [1] et émet l'hypothèse qu'un système à base de connaissances peut aider le personnel de la mesure à acquérir et à maintenir les connaissances nécessaires à la mesure fonctionnelle des logiciels. CosmicXpert a été conçu afin de vérifier cette hypothèse.

4.1 La méthode de mesure fonctionnelle COSMIC-FFP

4.1.1 Contexte

Le logiciel est une partie importante du budget des sociétés. Les organisations reconnaissent la nécessité de maîtriser leurs dépenses en logiciels et d'analyser les performances de leurs budgets alloués en développement logiciel et en maintenance, dans le but de tester leur performance face aux meilleurs dans ce domaine.

Les mesures sont donc nécessaires pour analyser la qualité et la productivité associées au développement et à la maintenance des logiciels. Des mesures techniques telles le nombre de lignes de code, sont les premières mesures acceptées et sont encore utilisées intensivement. Ces mesures servent à estimer les performances techniques des produits ou services selon le point de vue du développeur ainsi qu'à analyser l'efficacité et permettre des améliorations des performances de conception.

Mais ces mesures techniques ne sont pas suffisantes pour estimer la performance des produits ou services du point de vue de l'utilisateur et notamment, afin d'analyser la productivité. Des mesures fonctionnelles indépendantes des techniques de développement et d'implémentation doivent être utilisées afin de comparer la productivité des différentes techniques et technologies.

Une des mesures fonctionnelles inventées par Albrecht sont les points de fonctions dont l'objectif est d'estimer la taille du projet, exprimé en jour/homme à partir des besoins fonctionnels des utilisateurs. Cette mesure est toujours largement utilisée mais pose quelques problèmes lorsqu'elle n'est pas appliquée à des logiciels de gestion (logiciel en temps réel).

Ce problème d'adaptabilité avec les différents types d'application n'est évidemment pas la seule faiblesse des points de fonctions. Nous pouvons en citer d'autres qui ont trouvé une solution dans l'extension des points de fonctions :

- Les points de fonctions ne peuvent mesurer l'impact sur le logiciel de la taille des exigences dans toutes les couches et ne donnent aucune aide pour reconnaître les différentes couches. Un logiciel utilise de plus en plus de services « extérieurs » comme ceux fournis par un système d'exploitation ou encore un pilote de périphérique. Les fonctionnalités du logiciel sont dispersées dans plusieurs couches.
- Les points de fonctions sont incapables de définir la taille des exigences pour des composants « *peer* » dans une architecture multiniveaux.
- La complexité de la formule d'estimation et des ajustements pose parfois problème, entraînant des erreurs d'estimation variables.

Suite aux problèmes d'incapacité de mesurer universellement tous les logiciels rencontrés et les autres problèmes liés aux points de fonctions traditionnels, une méthode de points de fonctions étendus émanant du laboratoire de génie du logiciel et du laboratoire de métrique de l'UQAM est proposée en septembre 1997 afin de pallier à ces lacunes, tout en préservant les spécificités des points de fonctions traditionnels.

Les objectifs du projet COSMIC-FFP sont de développer, tester, faire connaître au marché et faire accepter de nouvelles méthodes d'étalonnage de logiciel pour supporter l'estimation et la mesure de performance.

La méthode COSMIC-FFP Functional Size Measurement a pour but de satisfaire les besoins :

- des fournisseurs de logiciel désireux de transformer les attentes des consommateurs en taille de logiciel à produire,
- des clients qui veulent connaître la taille fonctionnelle du logiciel livré pour l'utiliser comme une composante importante de mesure de la performance du fournisseur.

4.1.2 La méthode de mesure

La méthode de mesure COSMIC-FFP consiste à appliquer un ensemble de règles et de procédures sur un logiciel donné, tel qu'il est perçu par ses utilisateurs, c'est à dire à travers les besoins fonctionnels. Le résultat de l'application de ces règles et procédures est un nombre représentant la taille fonctionnelle du logiciel.

La méthode de mesure de COSMIC-FFP a été construite pour être indépendante des décisions d'implantation incorporées dans les artefacts opérationnels du logiciel que l'on veut mesurer. Pour atteindre cet objectif, la mesure est appliquée à un modèle générique de logiciel sur lequel les artefacts du logiciel à mesurer sont mis en correspondance. Ce processus est représenté à la Figure 4.1, tirée de [2].

Le modèle du processus de mesure COSMIC-FFP montre que, préalablement à l'application des règles et procédures de mesure, le logiciel doit être mis en correspondance avec un modèle de logiciel propre à COSMIC-FFP qui capture les concepts, définitions et relations (structures fonctionnelles) requises pour compléter l'exercice de la mesure de la taille fonctionnelle. Ce modèle est basé sur la notion de « *fonctionnalité utilisateur requise* » (FUR) indiquant la fonctionnalité livrée par les logiciels à ses utilisateurs.

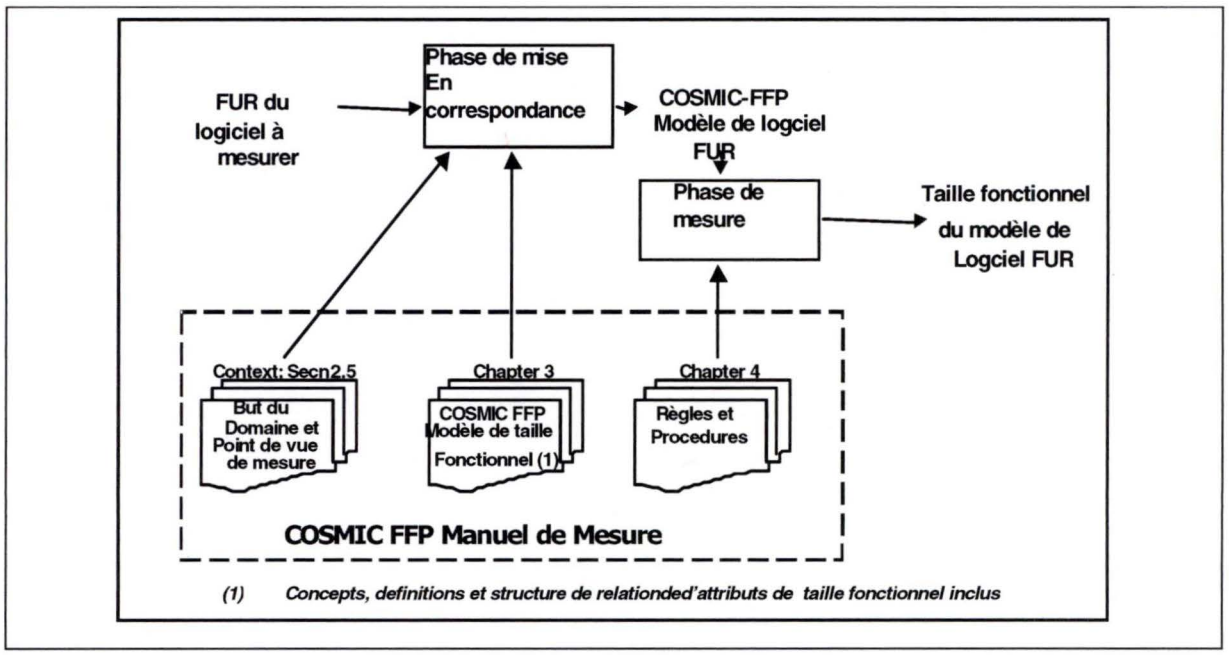


Figure 4.1 : *Modèle du processus de mesure de COSMIC-FFP*

Une fois le logiciel mis en correspondance avec le modèle de logiciel COSMIC-FFP, la mesure est réalisée en appliquant un ensemble de principes et de règles sur le modèle. On obtient ainsi un nombre représentant la taille fonctionnelle du modèle de logiciel. Par convention, ce nombre est ensuite utilisé pour représenter la taille fonctionnelle du logiciel lui-même.

4.1.2.1 Extraction des fonctionnalités utilisateurs requises (FUR)

En pratique, les FUR ne sont pas toujours présentes dans un document spécifique à l'organisation mais il est possible de les extraire d'autres artefacts ou individus connaissant bien le logiciel. Ces extractions sont effectuées soit :

- avant que le logiciel ne soit construit (architecture et conception)
- après que le logiciel soit construit

Les deux figures suivantes, tirées de [2], illustrent ces mécanismes d'extraction des FUR.

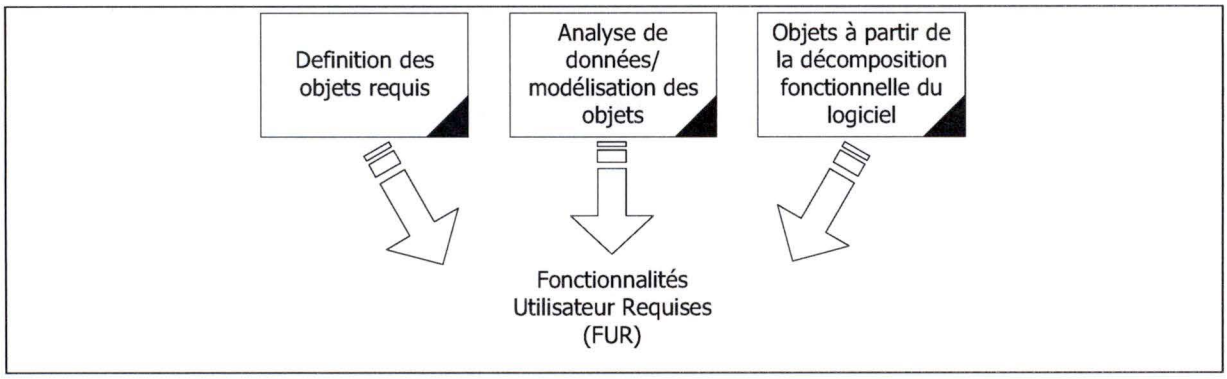


Figure 4.2 : *Le modèle des FUR avant implantation de COSMIC-FFP*

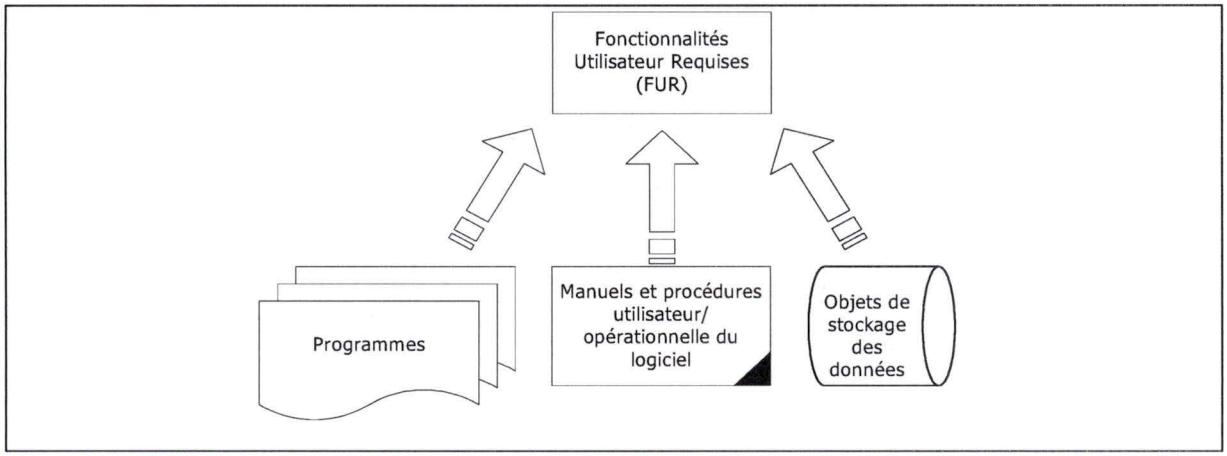


Figure 4.3 : *Le modèle des FUR après implantation de COSMIC-FFP*

4.1.2.2 Phase de mise en correspondance de COSMIC-FFP

La phase de mise en correspondance de COSMIC-FFP reçoit en entrée les artefacts d'un logiciel et, en utilisant un ensemble de règles et procédures définies, produit un modèle de logiciel spécifique (le modèle de logiciel COSMIC-FFP) pour faciliter la mesure de la taille fonctionnelle du logiciel. Le modèle de logiciel réalisé correspond à un sous-ensemble des FUR et fait partie d'une instance spécifique d'une méthode de la mesure de la taille fonctionnelle. Les règles et méthodes utilisées dans le processus de mise en correspondance du modèle de logiciel générique COSMIC-FFP sont résumées dans la Figure 4.4 issue de [2].

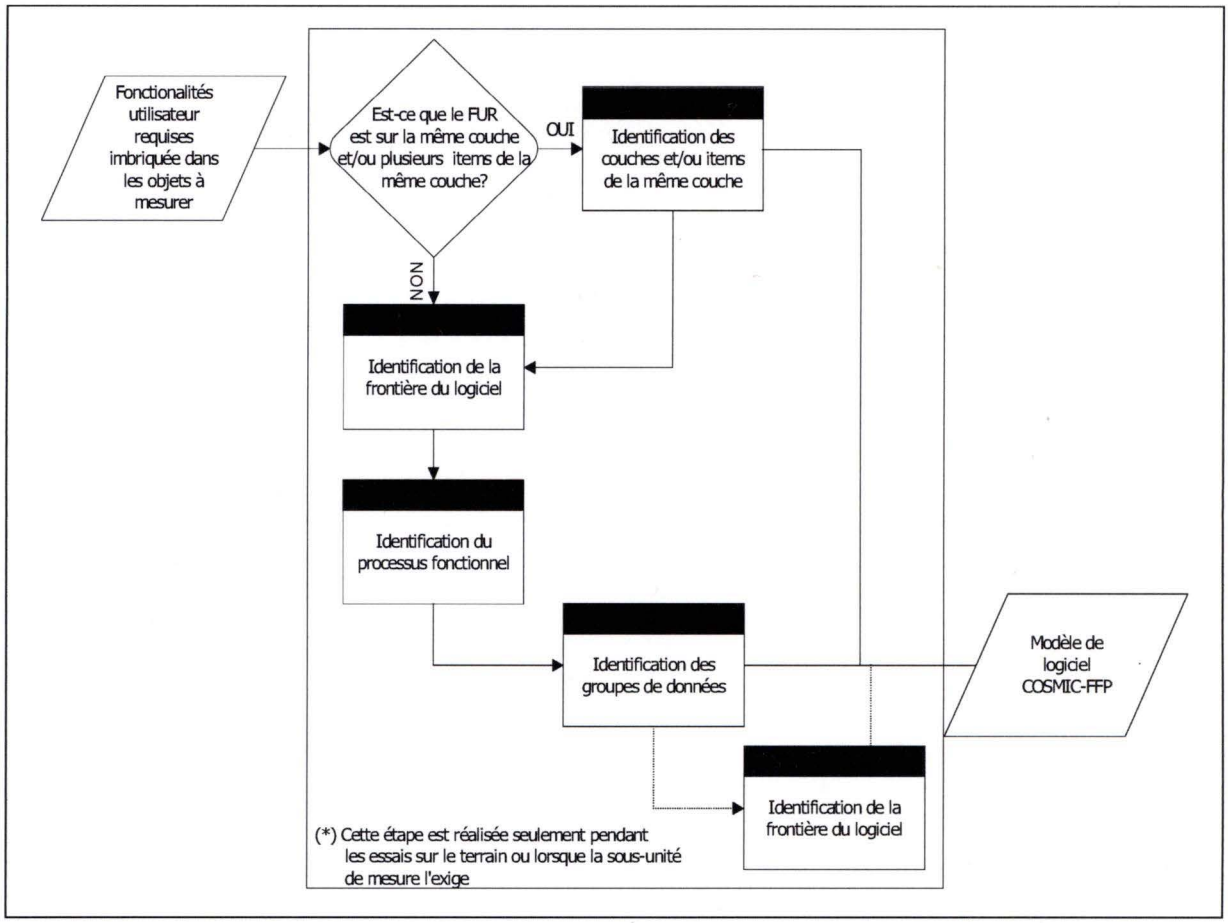


Figure 4.4 : *Processus de mise en correspondance COSMIC-FFP*

4.1.2.3 Phase de la mesure COSMIC-FFP

La phase de mesure de COSMIC-FFP reçoit en entrée une instance du modèle de logiciel de COSMIC-FFP et, en utilisant un ensemble défini de règles et de procédures, produit un nombre dont la magnitude est directement proportionnelle à la taille fonctionnelle du modèle. Ceci se base sur le principe que la taille fonctionnelle du logiciel est directement proportionnelle au nombre de ses mouvements de données élémentaires.

4.2 Les objectifs de CosmicXpert

CosmicXpert a pour objectifs :

- de faciliter et améliorer l'apprentissage de la méthode de mesure COSMIC-FFP ;
- d'améliorer la qualité des résultats de mesure ;
- d'améliorer les performances du mesureur.

En référence aux catégories de Waterman exposées dans le chapitre 1, on peut affirmer que CosmicXpert est un système expert de type « diagnostique » car il aide le mesureur à créer un modèle du logiciel devant être mesuré à partir des concepts de la méthode de mesure. Les artefacts du processus de développement du logiciel sont ici considérés comme les symptômes qui permettent de déterminer la taille du logiciel et ainsi d'identifier tous les concepts nécessaires à l'élaboration du modèle à partir duquel la mesure sera établie.
CosmicXpert peut également être classé dans la catégorie « interprétation » car une part du travail de la mesure est d'interpréter les données brutes constituant les artefacts du processus de développement et d'en dégager les concepts liés à la méthode de mesure. L'approche par diagnostique permet alors de confirmer ou d'infirmer l'interprétation faite des données brutes.
CosmicXpert peut enfin être considéré comme un système expert de type « éducation », puisque l'utilisation de CosmicXpert a une influence sur l'apprentissage de la méthode de mesure COSMIC-FFP par la présence d'explications des tâches et des concepts.

4.3 Le processus de mesure

Comme nous l'avons vu dans la section consacrée à la présentation de la norme COSMIC-FFP, l'application de la méthode de mesure COSMIC-FFP inclus deux processus :

- faire correspondre le modèle de mesure avec le logiciel à mesurer,
- appliquer les règles de mesure au modèle du logiciel dérivé du premier processus.

La mesure d'un logiciel est ainsi équivalente à la résolution d'un problème spécifique. Les paramètres du problème sont les éléments du logiciel mesuré, par exemple les différents artefacts du processus de développement comme le schéma entité-relation. C'est seulement

une fois que les paramètres sont clairement identifiés et adéquatement interprétés que le problème peut être résolu en utilisant les règles appropriées.

La Figure 4.5, tirée de [24], représente le cheminement cognitif de la résolution d'un problème.

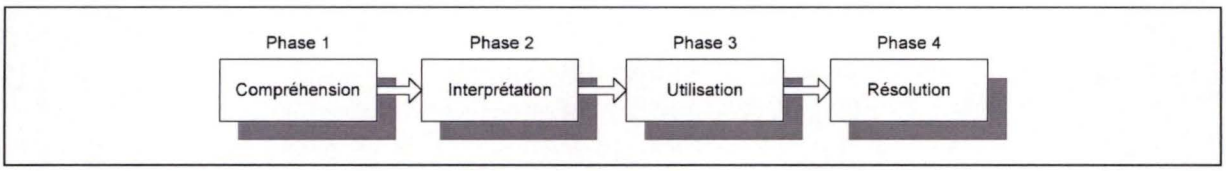


Figure 4.5 *Chemin cognitif du mesureur.*

4.3.1 Phase de compréhension

La phase de compréhension regroupe les activités permettant au mesureur de comprendre le problème. Desharnais et Abran [25] illustrent le chemin cognitif du mesureur par le problème suivant :

L'utilisateur désire consulter la base de donnée d'un fournisseur pour obtenir la liste des articles achetés durant le mois dernier auprès d'un fournisseur spécifique.

La mesure de la taille fonctionnelle de la fonctionnalité décrite nécessite la compréhension :

- des modalités du processus de publication de la liste ;
- les résultats du processus ou les groupes de données lus ;
- les validations et les résultats possibles des validations.

La qualité de la documentation du logiciel a une importance considérable dans cette étape de la résolution d'un problème de mesure.

Si parmi la documentation du logiciel, un schéma entité relation est présent, il permettra de définir les groupes de données utilisés pour obtenir la liste des articles achetés.

Pour que la compréhension du lien existant entre schéma entité-relation et groupes de données soit possible, il est nécessaire que le mesureur ait des connaissances dans le domaine du génie logiciel et dans le domaine des mesures fonctionnelles.

4.3.2 Phase d'interprétation

Durant la phase d'interprétation le mesureur doit identifier les artefacts du logiciel ayant un sens pour l'application de la méthode de mesure.
Dans la phase précédente le mesureur a compris qu'un lien existe entre les entités et les groupes de données. Dans cet exemple, la manière dont le mesureur interprète ce lien a une influence sur le nombre de groupes de données identifiés.

4.3.3 Phase d'utilisation

La phase d'utilisation consiste pour le mesureur à utiliser les règles décrites dans le manuel de mesure de COSMIC-FFP afin de s'assurer que son interprétation est correcte.
Le mesureur utilise les règles concernant l'identification d'un groupe de données pour vérifier son interprétation du lien existant entre les entités et le concept de groupes de données.
Les règles concernant les mouvements de données permettent de transcrire dans le modèle COSMIC-FFP le processus de publication de la liste et l'étape de validation de l'exemple décrit précédemment.

4.3.4 Phase de résolution

La résolution est la phase durant laquelle le mesureur doit faire appel à ses connaissances implicites en ce qui concerne le développement logiciel et les tâches du processus de mesure à réaliser pour résoudre le problème de mesure.
Solutionner le problème dans l'exemple revient à identifier les groupes de données utiles pour la production de la liste ainsi que les mouvements de données du processus y compris la validation.
L'exemple ici est simple car il n'y a effectivement qu'un seul processus fonctionnel. Dans des cas plus complexes, le problème de mesure est décomposé en plusieurs sous problèmes (par exemple, l'identification des processus fonctionnels constitue un premier problème, chaque processus fonctionnel est ensuite divisé en d'autres problèmes comme l'identification des groupes de données dans chacun des processus fonctionnels.)
Le mesureur doit utiliser ses connaissances et son expérience pour confirmer ou infirmer les solutions possibles issues d'interprétations différentes du problème.

4.4 Modélisation des connaissances

Pour comprendre le design de la base de connaissances de CosmicXpert, il est nécessaire de décrire différents types de connaissances. Dans cette section, nous nous référerons particulièrement à van Heijst cité dans [24]. Selon lui, il existe au moins cinq types de connaissances à prendre en compte :

- Les *tâches* : correspondent aux objectifs à atteindre dans la résolution du problème.
- Les *méthodes de résolution du problème* : sont des manières d'atteindre les objectifs décrits dans les tâches. Dans certaines représentations des connaissances, les méthodes de résolution définissent des sous-tâches auxquelles peuvent être associées de nouvelles méthodes de résolution.
- Les *inférences* : décrivent les étapes primitives de raisonnement dans le processus de résolution du problème. L'ensemble des inférences forme le modèle fonctionnel et est communément appelé le *modèle d'inférence*.
- Les *ontologies* : décrivent la structure et le vocabulaire du domaine de connaissances donné.
- Le *domaine de connaissances* : fait référence à une collection d'énoncés sur le domaine.

Les types de connaissances sont présentés à la Figure 4.6, également tirée de [24].

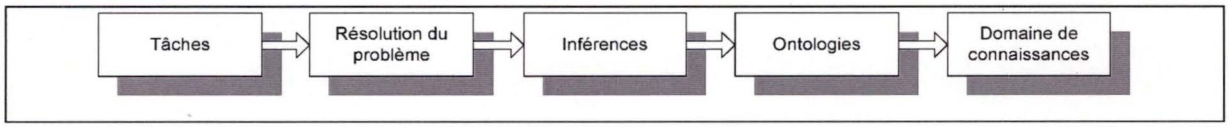


Figure 4.6 *Types de connaissances*

De façon à permettre au lecteur un maximum de compréhension des termes utilisés, nous allons commencer cet exposé des types de connaissances par les ontologies. Nous poursuivrons ensuite selon l'ordre préconisé par van Heijst.

4.4.1 Ontologies

Selon Grüber cité dans [24], une ontologie est « *une spécification explicite d'une conceptualisation. Le terme est emprunté à la philosophie, où l'ontologie est un compte-rendu*

systématique de l'Existence. Pour les systèmes d'Intelligence Artificielle, ce qui existe correspond à ce qui peut être représenté. »

Pour construire le système expert, il est utile de sélectionner une ontologie du génie logiciel autour de laquelle le plus grand consensus s'est formé. Tel que mentionné dans [24], le système a été construit « *pour fournir une caractérisation consensuellement validée de la frontière de la discipline de l'ingénierie logicielle et pour fournir un accès au « Body of Knowledge » supportant cette discipline* ». C'est pourquoi CosmicXpert utilise les définitions et la structure adoptées dans SWEBOK (*Software Engineering Body of Knowledge*).

D'un autre côté, la méthode de mesure à laquelle le système à base de connaissances sera appliqué, est la méthode de mesure COSMIC-FFP. Celle-ci faisant d'ailleurs l'objet d'une norme internationale émise par l'*International Standard Oraganization* (ISO). Cette méthode possède un ensemble de concepts, de définitions et de principes propres et bien documentés. Ces concepts sont repris à la Figure 4.7.

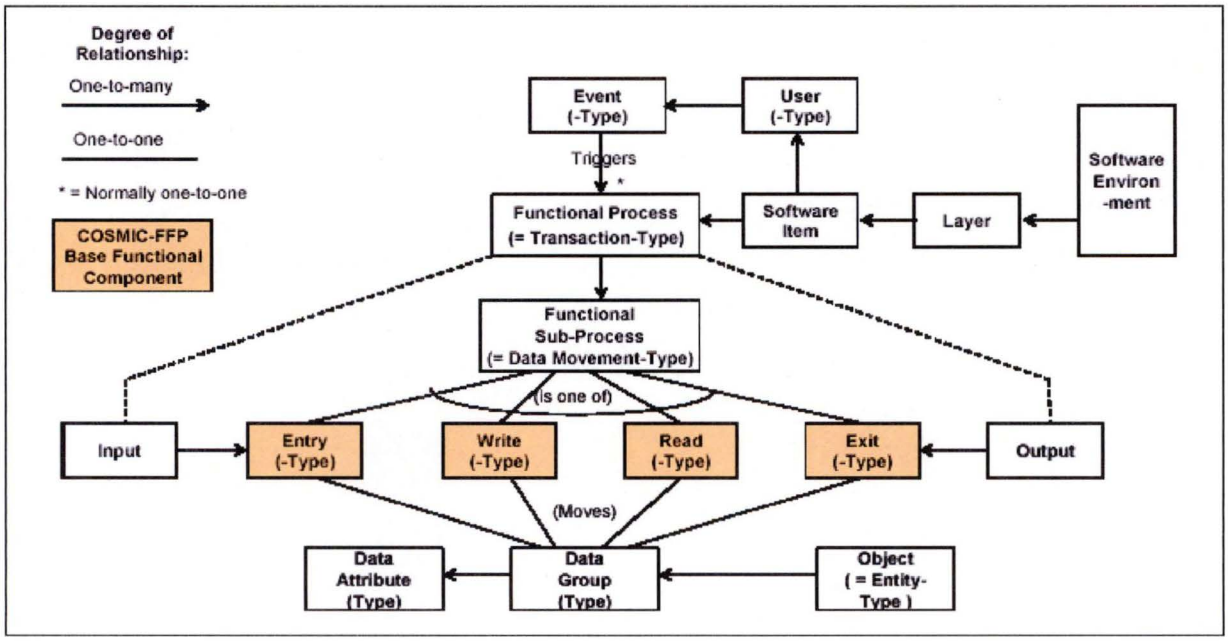


Figure 4.7 *Ensemble des concepts de COSMIC-FFP.*

L'ontologie du système expert fait le lien entre les deux précédentes. La Figure 4.8 fournit un diagramme entité-association des concepts de l'ontologie de CosmicXpert.

La définition des entités de ce diagramme est indispensable pour la compréhension de CosmicXpert.

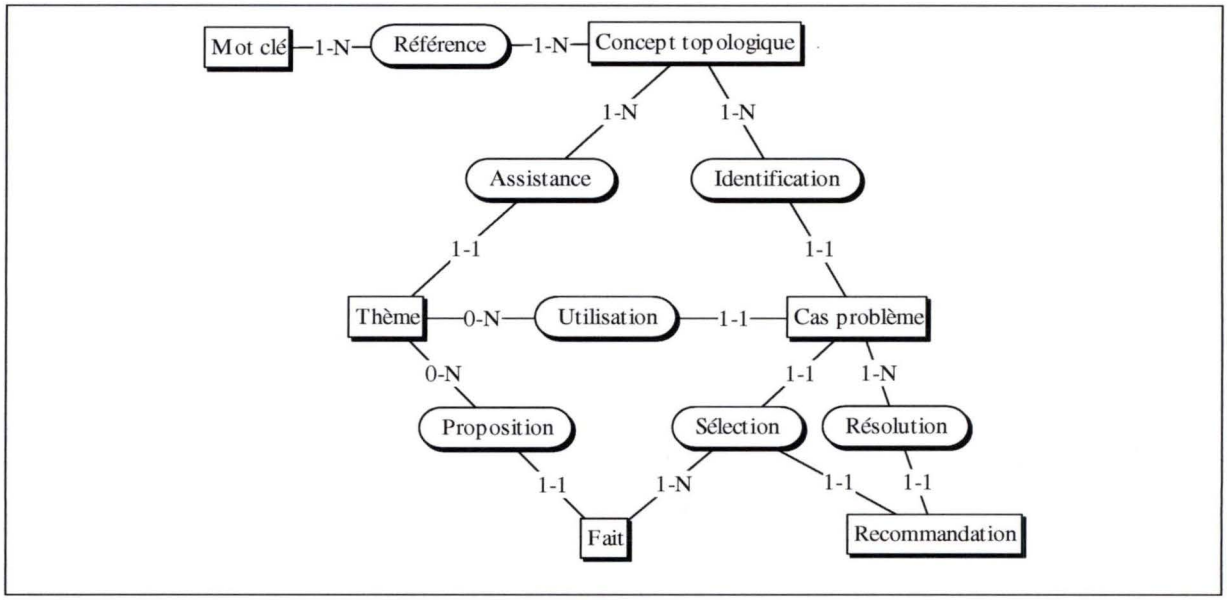


Figure 4.8 *Ontologie de CosmicXpert*

Les *concepts topologiques* sont des concepts qui servent à établir des liens entre les concepts de l'ingénierie du logiciel et ceux de COSMIC-FFP. Ils sont définis dans l'ontologie de COSMIC-FFP.

Un *mot-clé* est n'importe quel mot du domaine du génie logiciel ou de COSMIC-FFP pouvant être associé à un concept topologique.

Un *cas problème* est un cas pour lequel le mesureur doit utiliser un processus de diagnostic pour identifier correctement un concept de l'ontologie COSMIC-FFP.

Les *thèmes* sont les pistes de réflexion à suivre par le mesureur pour trouver une solution aux cas problèmes.

Un *fait* est le constat du mesureur par rapport à un thème et au cas problème relié à ce thème.

Une *recommandation* est la proposition de solution fournie par le système ou des indications permettant au mesureur de raffiner sa solution.

4.4.2 Les tâches

La tâche indique les buts du mesureur et les stratégies employées pour réaliser ces buts. Pour le mesureur, l'objectif final est d'identifier les différentes instances des concepts de la méthode de mesure COSMIC-FFP à partir des artefacts de la documentation du logiciel (ou des commentaires des développeurs).

Le Tableau 4.1 présente l'ensemble des tâches détaillées intégrées à CosmicXpert pour aider le mesureur à atteindre son objectif.

Tableau 4.1 *Description des tâches du mesureur pour CosmicXpert*

No.	Tâche	Description
1.	Entrer un mot-clé	Le mesureur entre un mot-clé qui aidera le système expert à trouver des concepts topologiques.
2.	Chercher un concept topologique	Le système expert présente une liste de concepts topologiques au mesureur.
3.	Donner un ordre de priorité aux concepts topologiques	Le système expert présente les concepts topologiques selon une certaine priorité en fonction du mot clé entré.
4.	Choisir un concept topologique	Le mesureur choisit un concept topologique.
5.	Trouver les cas problèmes	Le système expert recherche les cas problèmes reliés au concept topologique choisi par le mesureur.
6.	Donner un ordre de priorité aux cas problèmes	Le système expert ordonne les cas problèmes selon une priorité définie par le concept topologique choisi.
7.	Choisir un cas problème	Le mesureur choisit un cas problème correspondant à son interprétation du problème.
8.	Afficher les thèmes	Le système expert affiche les thèmes utiles à la résolution du cas problème.
9.	Répondre aux thèmes	Le mesureur choisit un fait pour chaque thème proposé.
10.	Afficher le résultat	Un pourcentage est présenté au mesureur, il est calculé en fonction des faits choisis pour les thèmes.
11.	Evaluation du résultat	Le système expert évalue le résultat basé sur des heuristiques.

12.	Recommandation/explication	Le système expert recommande soit une solution pour le cas problème, soit un autre cas problème et/ou une explication pour le cas problème non résolu.
13.	Accepter la recommandation	Le mesureur décide si la recommandation est acceptable.
14.	Choisir un cas problème (nouveau)	Le mesureur choisit un autre cas problème, que ce soit un cas suggéré par le système expert ou un autre.

4.4.3 Méthode de résolution de problèmes

Une méthode de résolution de problèmes est assignée à chaque tâche du processus de mesure. Les principales méthodes de résolution de problèmes sont les suivantes :

- la recherche par mots-clés : à partir du vocabulaire du génie logiciel et de celui de la mesure fonctionnelle COSMIC-FFP, des mots-clés seront retenus et utilisés pour permettre au mesureur d'identifier l'environnement du problème ;
- la déduction : par exemple, à partir de cas problèmes précis, le système de connaissances déduira les thèmes à utiliser pour aider à le résoudre ;
- l'induction : par exemple, à partir des réponses aux thèmes, le système de connaissances induira si le concept topologique identifié par le mesureur est le bon.

4.4.4 Inférences

Les inférences sont les étapes de raisonnement les plus fines d'une méthode de résolution de problèmes. Le mesureur effectue deux types d'inférences :

- des recherches permettant de prioriser (avec des %) les cas problèmes à résoudre à partir de mots-clés ou d'autres informations ;
- des recherches permettant de solutionner les cas problèmes ou de recommander d'autres tâches pour mieux circonscrire les cas problèmes à partir de calculs basés sur la théorie de la certitude.

4.4.5 Le domaine de connaissance de CosmicXpert

La connaissance du domaine est un ensemble d'énoncés sur le domaine. Ceux-ci proviennent du mesureur ou de l'expert du domaine de la mesure fonctionnelle pour réaliser la mise en correspondance entre les artefacts du logiciel à mesurer et les règles de la méthode de mesure. Ainsi, le mesureur élabore un énoncé pour mettre en correspondance le concept de processus dans COSMIC-FFP et le modèle de processus dans une méthodologie ou un corpus de connaissances en génie logiciel. Il lui faudra trouver un mot-clé résumant cet énoncé. L'expert suggère les mots-clés mais il appartient au mesureur de choisir selon sa convenance.
Cet énoncé (mot-clé) permet d'identifier un concept topologique qui s'appuie sur l'ontologie de COSMIC-FFP. Le concept topologique permet la mise en correspondance des concepts des méthodes de développement et des concepts de la méthode de mesure COSMIC-FFP. L'expert décide des liens (et leur force) entre le mot-clé et le concept topologique.

4.5 Un système expert hybride

La Figure 4.9, tirée de [24] définit l'enchaînement des tâches décrites dans le Tableau 4.1. Cet enchaînement de tâches définit le processus suivi par un mesureur dans CosmicXpert afin de bien interpréter les règles de mesure de COSMIC-FFP.
Ce schéma permet également de mettre en évidence le côté hybride de CosmicXpert. La première partie du processus (figures avec fond blanc) définit une approche du type raisonnement par cas (cf. chapitre 1) tandis que la seconde partie (figures avec fond gris) définit une approche ressemblant à un système basé sur des règles.

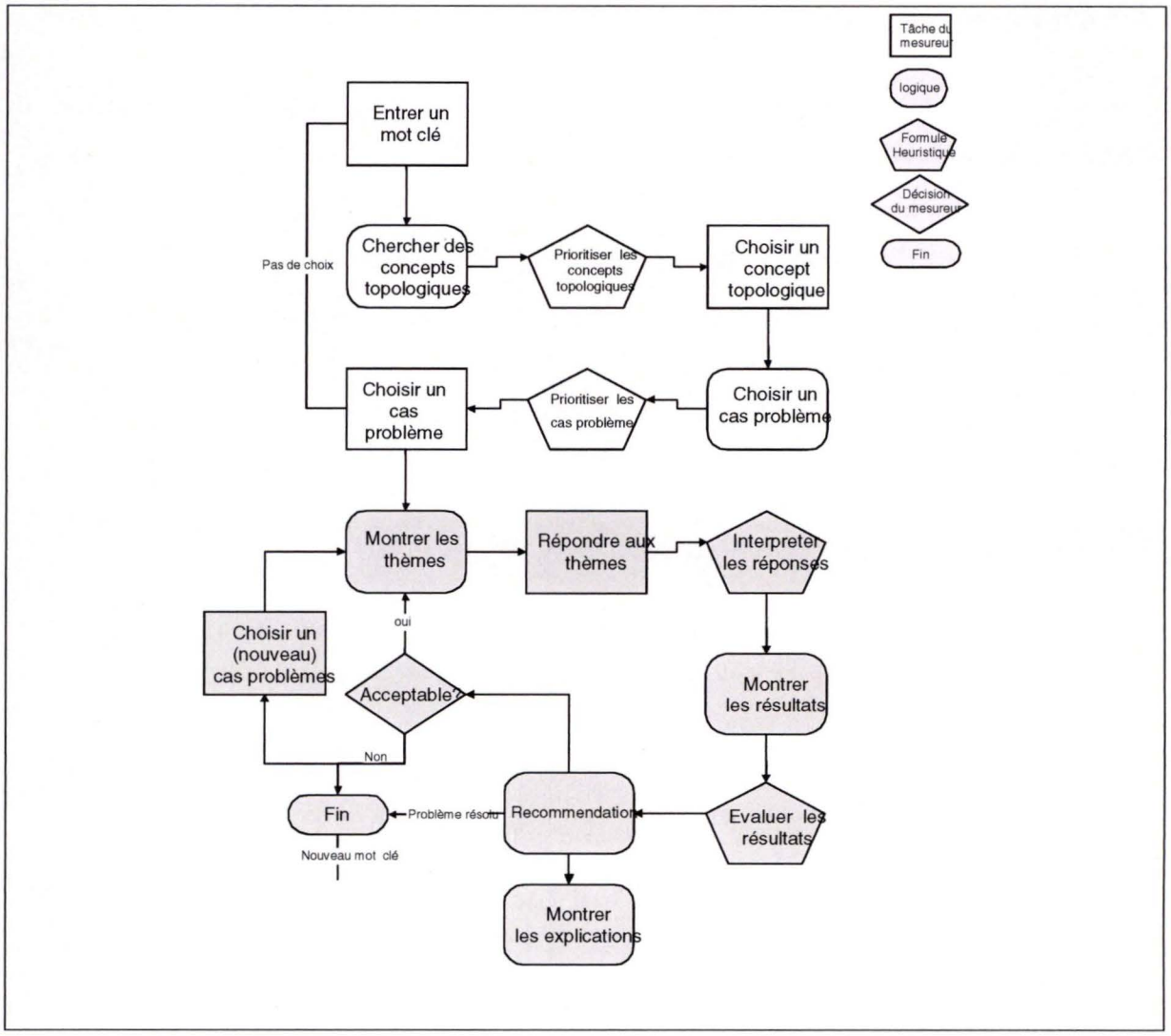


Figure 4.9 *Enchaînement des tâches de la mesure dans le système expert.*

4.5.1 CosmicXpert : un système expert basé sur des cas

Les premières étapes de l'enchaînement des tâches décrit ci-dessus utilisent les principes d'un raisonnement par cas (cf. chapitre 1).

Selon Kolodner [10] pour qu'un système de raisonnement par cas soit efficace dans un domaine de connaissance, il faut avoir l'intuition que certaines situations reviennent avec régularité dans ce domaine.

C'est le cas du domaine de COSMIC-FFP. Il est en effet possible d'utiliser des cas problèmes déjà résolus pour en résoudre de nouveaux. Le nombre de règles dans COSMIC-FFP étant relativement limité, les options de solutions pour les différents cas problèmes sont également limitées. De plus ce sont les mêmes règles qui s'appliquent à tous les cas problèmes.

4.5.1.1 Définition d'un cas

Le concept de cas problème présent dans CosmicXpert respecte la définition d'un cas :

> « *Un cas est un morceau de connaissance pris dans le contexte. Le cas représente une expérience qui enseigne une leçon fondamentale pour réaliser les objectifs de la personne qui l'étudie* » [10]

4.5.1.2 Identification des cas

Les cas problèmes constituant la base de connaissances de CosmicXpert sont construits à partir d'étude de cas. Il faut faire la distinction entre une *étude de cas* et un *cas problème*.
Une *étude de cas* est constituée de documents expliquant un objet à mesurer, par exemple le logiciel contrôlant un cuiseur de riz.
Un *cas problème* est un sous-ensemble d'une étude de cas. Par exemple dans l'étude de cas du cuiseur de riz, presser sur le bouton « démarrer » constitue un cas problème. Le problème est de savoir si presser sur le bouton « démarrer » est une entrée au sens de COSMIC-FFP.
La base de CosmicXpert est construite à partir de plusieurs études de cas. Celles-ci sont représentatives du domaine de la gestion des affaires et du domaine du temps réel. Les cas problèmes extraits de ces études de cas doivent couvrir les principaux concepts de la méthode COSMIC-FFP.
Pour sélectionner les cas problèmes parmi tous ceux que soulève une étude de cas, il est utile de s'appuyer sur certains des principes que l'on retrouve dans [10]:

- un cas problème enregistre des expériences qui sont différentes à chaque fois. Cependant, toutes les différences ne sont pas importantes à enregistrer. Les cas qui valent d'être enregistrés sont ceux qui donnent une leçon utile ;
- les leçons utiles sont celles qui ont la possibilité d'aider une personne à réaliser un but ou un ensemble de buts plus facilement dans le futur ou qui avertissent de la possibilité d'un échec ou encore qui mettent en évidence un problème imprévu.

4.5.1.3 Contenu d'un cas

Un autre principe qu'énonce Kolodner permet de déterminer ce qui est nécessaire dans un cas problème : « un cas problème vient selon différentes formes et différentes tailles, couvrant une durée relativement grande ou petite, associant des solutions avec des problèmes, des résultats avec des situations, ou les deux. »

Dans le cadre de CosmicXpert un cas problème doit contenir la description du problème spécifique à la mesure et les solutions (recommandations) associées à ce problème. Pour associer les solutions aux problèmes, CosmicXpert utilise une approche par règle décrite ci-dessous.
Le cas problème doit également fournir une description de son contexte, autrement dit de l'étude de cas à partir de laquelle il a été conçu.

4.5.2 CosmicXpert : un système expert basé sur des règles

L'enchaînement des tâches décrit dans la Figure 4.9 se termine par une approche par règles. L'objectif de cette approche dans CosmicXpert est d'aider le mesureur à appliquer la solution d'un cas problème existant à son problème. Une fois le cas problème existant sélectionné, le mesureur vérifie si les règles de COSMIC-FFP sont respectées dans le cas problème existant et peut ainsi appliquer les règles dans le cas qui le concerne.
Afin d'aider le mesureur à appliquer les règles nécessaires à l'identification d'un concept topologique, le système propose des thèmes liés aux règles, sous la forme de questions. A chaque thème est associé une série de faits, c'est-à-dire des propositions de réponse.
Une fois que le mesureur a choisi des faits pour les différents thèmes, le système propose une recommandation. La recommandation est sélectionnée en fonction d'une heuristique utilisant la théorie de la certitude. Cette recommandation permet au mesureur d'évaluer la solution du cas problème et ainsi de l'accepter ou de la refuser.

4.6 Description du premier prototype

Comme précisé en début de ce chapitre, CosmicXpert est l'aboutissement de la thèse de doctorat de Jean-Marc Desharnais. Ce système a été conçu afin de vérifier l'hypothèse selon laquelle un système à base de connaissances peut aider le personnel de la mesure à acquérir et à maintenir les connaissances nécessaires à la mesure fonctionnelle des logiciels.
La création de ce système ne s'est pas faite en une seule étape, mais est le fruit de plusieurs tentatives, donnant naissance à trois prototypes.
Le premier prototype fait l'objet de cette section, les prototypes suivants seront exposés dans les chapitres 4 et 5.

4.6.1 Introduction

Le premier prototype a été réalisé, en collaboration avec Mr Desharnais, par Tim Küssing dans le cadre de son travail de fin d'études [3]. L'objectif de ce prototype était d'implémenter toutes les fonctionnalités nécessaires à la vérification de l'hypothèse de départ.
Pour ce faire, le domaine d'application, les acteurs et les cas d'utilisation du système ont été identifiés. Ils sont présentés dans le point suivant.

4.6.2 Identification des cas d'utilisation

La Figure 4.10 présente le diagramme des cas d'utilisation de CosmicXpert. Trois utilisateurs différents du système ont été identifiés :

- L'*expert* est l'acteur responsable du contenu de la base de connaissances. Il participe aux cas d'utilisation concernant la gestion de la base de connaissances.
- Le *mesureur* est l'utilisateur pour qui le système est conçu. L'objectif du mesureur est de résoudre un problème de mesure fonctionnelle en effectuant les tâches décrites dans la Figure 4.9.
- L'*administrateur* a un rôle réduit à la gestion des accès au système. Il détermine qui parmi les utilisateurs potentiels joue le rôle d'expert ou de mesureur.

Les cas d'utilisation du premier prototype de CosmicXpert sont présentés dans l'Annexe 3.

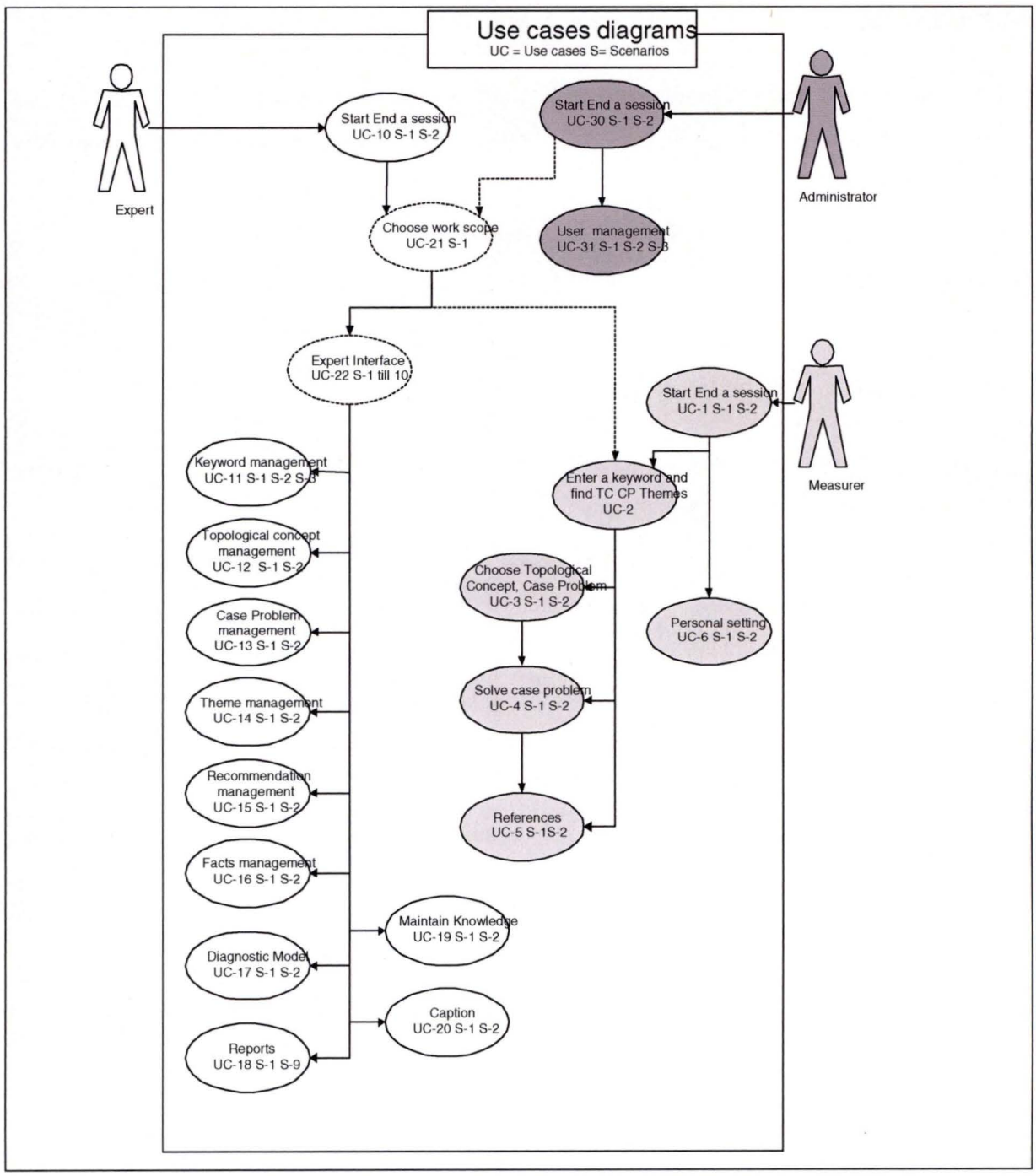


Figure 4.10 *Diagramme des cas d'utilisation du premier prototype de CosmicXpert.*

4.6.3 Architecture du premier prototype

Le premier prototype de CosmicXpert, comme le montre la Figure 4.11, s'appuie sur l'utilisation des technologies Visual Basic (VB6) et Microsoft ACCESS. De façon plus générale, on peut constater que le système est construit sur base de la séparation entre

l'application et la base de données. Un fichier *Universal Data Link* (UDL) assure le lien entre les deux parties.

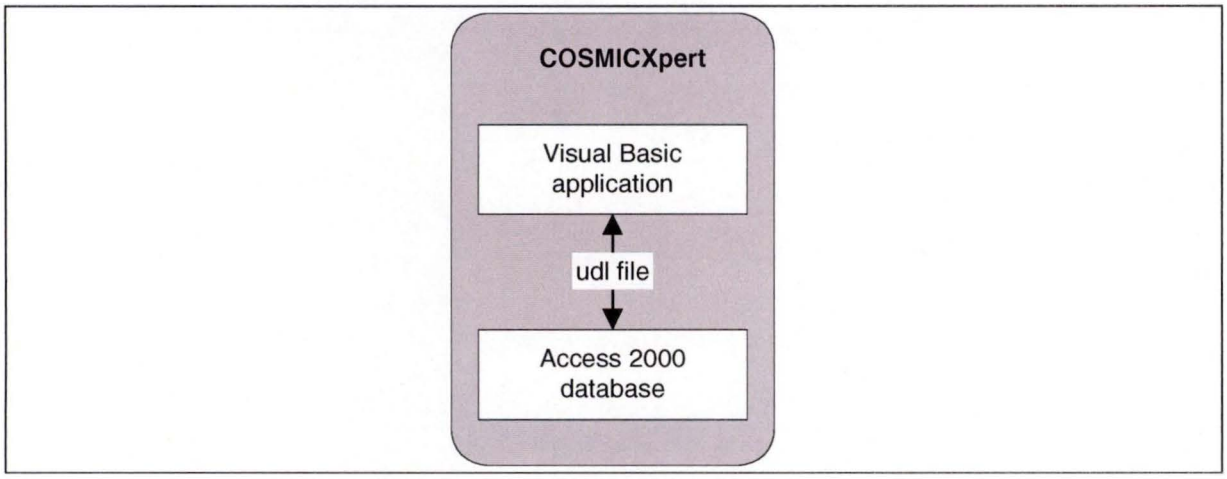


Figure 4.11 *Composition du premier prototype de CosmicXpert*

En plus de ces deux niveaux, il existe un ensemble important de fichiers au format *Rich Text Form* (RTF) contenant les « *connaissances*[7] »du système. Il s'agit par exemple des définitions des différents concepts, des recommandations qui sont présentées aux mesureurs,...

4.6.4 Résultats du premier prototype

Le premier prototype de CosmicXpert répond partiellement à ces exigences initiales. En effet, le système doit permettre de vérifier qu'un système à base de connaissances est en mesure d'aider le personnel de la mesure à acquérir et à maintenir les connaissances nécessaires à la mesure fonctionnelle de logiciels.

En se référant à la Figure 4.10, ainsi qu'à l'Annexe 3, on peut constater que les fonctionnalités nécessaires pour atteindre cet objectif ont été mises à la disposition du personnel de la mesure. Cependant, la conception du premier prototype n'a pas pris en compte dans les besoins de l'expert, la facilité de vérifier et de valider le contenu de la base de connaissances.

Les difficultés que pourrait rencontrer un expert qui se risquerait à ce genre de vérification sont aisées à comprendre, au regard du nombre impressionnant des fichiers qui composent la base de connaissances du premier prototype. En effet, il y existe plus de 800 fichiers RTF, dont la majorité des connaissances qui les composent sont redondantes.

---

[7] Le terme de *connaissances* n'est pas le plus approprié ici, mais les documents RTF dont nous faisons mention sont à l'origine des connaissances utilisées par les prototypes suivants.

Or, pour que CosmicXpert satisfasse pleinement ces objectifs, il est nécessaire que la maintenance des connaissances du système soit assurée.

Nous voyons poindre ici les exigences du second prototype. Ce dernier est présenté dans le chapitre suivant.

Chapitre 5 : Description du second prototype

Tel que mentionné dans le chapitre précédent, les faiblesses du premier prototype de CosmicXpert font apparaître le problème de la vérification et de la validation.
L'objectif principal du second prototype est de répondre à ces faiblesses. De plus, ce dernier réalise le passage à un environnement distribué.
Ces caractéristiques sont expliquées en détail dans ce chapitre.

5.1 CosmicXpert, une application distribuée

Avant de nous pencher sur le problème de la vérification et de la validation de la base de connaissances de CosmicXpert, arrêtons nous sur la première différence majeure entre le premier et le second prototype réside dans le changement d'architecture.

Alors que le premier prototype est basé sur une application Visual Basic (VB6) et sur une base de connaissances composée d'une base de données ACCESS et de fichiers RTF, le tout travaillant de façon isolée sur une seule machine, le second prototype, quant à lui, est constitué d'une application distribuée, permettant à plusieurs mesureurs à travers le monde de consulter la même base de connaissances, via leurs navigateurs Internet. Les technologies utilisées pour l'implémentation de l'application sont *Java Server Page* (JSP), HTML (*Hyper Text Markup Language*) et Java. La base de connaissances utilise des fichiers XML, nous reviendrons plus en détail sur cette dernière technologie dans les sections suivantes.

Le changement de contexte représente une certaine difficulté. Il implique en effet de repenser totalement l'implémentation du système. Devant la charge de travail que cette étape nécessite, seule l'interface du mesureur a été réalisée.
Mais il s'agit également d'une opportunité, puisque la vérification et la validation de la base de connaissances vont pouvoir être faites de manière semi-automatisée, grâce au choix judicieux des technologies, comme nous le verrons dans la section suivante.

5.2 Structuration de la base de connaissances

La base de connaissances du premier prototype est constituée d'une base de donnée ACCESS et de fichiers RTF regroupant des connaissances peu structurées. Les documents de spécification du système expert décrivent la structure de la base de données mais pas la structure du contenu des fichiers RTF.
Dès lors, François Gruselin et Julien Vilz [4], auteurs du second prototype, ont commencé la vérification en construisant une charte de structuration spécifiant entre autres la structure du contenu des fichiers RTF.

La charte de structuration, présentée dans l'Annexe 1, permet d'identifier les différentes connaissances de la base de connaissances et de résoudre les différentes ambiguïtés concernant la structure du format et du contenu des connaissances. Cette charte est le résultat d'un processus de rétroingénierie des documents RTF. Elle est construite en analysant un échantillon des connaissances afin de rassembler leurs informations et d'identifier une allure générale de chaque concept de la base de connaissances de CosmicXpert (concept, mot-clé, concept topologique, cas problème, thème, recommandation).

La mise au point d'un processus de vérification et de validation de la base de connaissances de CosmicXpert a révélé de nombreuses anomalies au sein de la base de connaissances du prototype 1. La conclusion que Gruselin et Vilz tirent après la première inspection du premier prototype est la suivante :

> *« En conclusion, suite à l'expérimentation, nous avons constaté un nombre impressionnant d'erreurs (817 erreurs), 523 images redondantes et environ 757 redondances internes entre les connaissances, soit près de 2097 erreurs. Ces erreurs s'expliquent par le fait que nous effectuons notre première vérification, que les connaissances n'ont pas suivi un processus de construction rigoureux (pas de spécification pour les fichiers RTF) entraînant des erreurs de cohérence, de syntaxe, de présentation et de structuration. Le format RTF introduit énormément de redondances entre les connaissances. »*[4]

5.3 Le format des fichiers

La solution proposée dans le deuxième prototype, est une solution technique permettant d'automatiser le processus de vérification et de faciliter la correction des erreurs de cohérence et de structuration. Cette solution consiste en l'utilisation d'un nouveau format pour représenter les connaissances. Ce nouveau format est une norme reconnue XML (*eXtensible Markup Language*) vérifié par des fichiers XSD (*XML Schema Definition*) définissant la structure des fichiers XML. L'ensemble de ces normes sont définies par le *World Wide Web Consortium* (W3C).

5.3.1 XML

Le format XML a été défini à l'origine pour la publication à large échelle de documents électroniques. Une série d'autres normes accompagnent le format XML avec des objectifs différents. Trois autres normes sont intéressantes dans le cas de vérification et validation de la base de connaissances.
XML est un méta langage facilitant la création de langage. Chaque schéma XML définit un nouveau langage basé sur le méta langage XML.

5.3.2 XML Schema

Le schéma XML est une norme qui est toujours en développement mais on trouve déjà sur le marché beaucoup d'outils qui permettent de l'utiliser. Cette norme permet de décrire la structure, le contenu et la sémantique d'un document XML. Elle se matérialise dans les fichiers XSD (*XML Schema Definition*) attachés aux fichiers XML.
C'est grâce à cette norme que la vérification de la structure des documents composant la base de connaissances va pouvoir être automatisée.

5.3.3 XSL

XSL (*eXtenslible Stylesheet Language*) est une famille de langages permettant de décrire comment doit être affiché un document XML d'un certain type. Le type de document étant défini par un schéma XML.

La famille de langages XSL comprend la norme X-path pour référencer les éléments dans un document XML et XSLT un langage définissant la transformation de fichiers XML.

5.3.4 XHTML

XHTML est une reformulation de HTML qui respecte le méta langage XML et qui peut être interprété par un navigateur Internet.
Afin de conserver la structuration des textes contenus dans les documents, un sous-ensemble du langage XHTML est utilisé. Il permet de définir la structuration en paragraphes, en listes et l'emplacement des images à insérer.
Un sous-ensemble du langage XHTML sera utilisé dans tous les types de documents.
Le langage de présentation XSL peut produire une vue en XHTML pouvant être utilisée dans la plupart des navigateurs.

5.3.5 Combinaison des langages

Pour résumer, un fichier XML contient les informations du document RTF structurées de sorte que le fichier XML puisse être vérifié par le schéma XML correspondant à son type de document.
Une procédure écrite en XSL permet de créer une vue du ficher XML en le transformant en XHTML.
La Figure 5.1, tirée de [4], illustre la combinaison des langages.

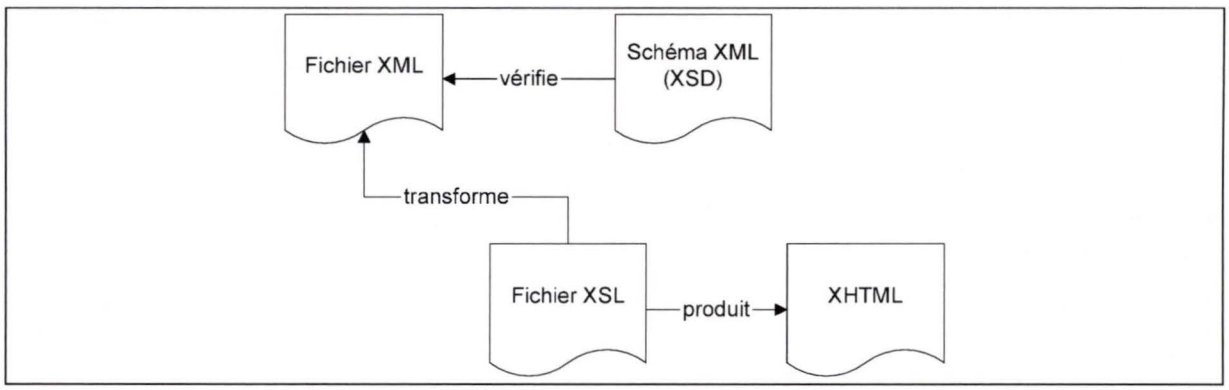


Figure 5.1 *Combinaison des langages*

5.4 Résultats obtenus

Comme le précisent Gruselin et Vilz, le processus de vérification et de validation de la base de connaissances de CosmicXpert instauré dans le deuxième prototype a permis d'apporter :

- *« Une spécification de la structure des connaissances plus rigoureuse, à travers les schémas XML et la charte de structuration.*
- *Un nouveau format de fichier XML permettant une vérification automatique du respect des structures spécifiées pour les connaissances. Ce format permet également de dissocier la présentation des connaissances des connaissances proprement dites.*
- *Une réduction du nombre de fichiers, sans limiter le nombre de connaissances.*
- *Une cohérence des fichiers par rapport à la norme COSMIC-FFP et à son manuel de la mesure. Cette cohérence est, entre autres, facilitée car les définitions ne sont plus dupliquées dans les différents fichiers mais rassemblées dans un glossaire et liée aux connaissances.*
- *Une correction de la syntaxe (grammaticale et orthographique) et de la présentation de tous les documents.*
- *Une bonne complétude de l'ensemble de nos connaissances et de la base de connaissances elle-même.*
- *Une élimination de la redondance facilitant la maintenance des différentes connaissances et la création de nouvelles connaissances.* »[4]

Nous pouvons constater que le second prototype a permis de réaliser la vérification et la validation de la base de connaissances de CosmicXpert. Néanmoins, ce prototype souffre toujours de quelques faiblesses. En effet, le changement de format des fichiers de la base de connaissances, ainsi que le changement radical d'architecture (passage d'une application Visual Basic à une application Web) ont fait que le système ne satisfait plus à ces exigences originelles. Seules les fonctionnalités du mesureur ont été implémentées dans ce second prototype. Les fonctionnalités permettant à un expert de maintenir les connaissances du système, et les fonctionnalités propres à l'administrateur ayant été laissées en suspens.

Chapitre 6 : Présentation du troisième prototype

6.1 Les faiblesses de CosmicXpert

Outre les raisons théoriques développées dans le premier chapitre, d'autres critères font que nous nous sommes rendu compte qu'une vérification automatique de CosmicXpert dans son environnement actuel est plus que nécessaire. Cette section tente de les énumérer.

6.1.1 Règles de la mesure COSMIC-FFP

La théorie de la mesure par les points de fonctions, et plus particulièrement la méthode COSMIC-FFP est en constante évolution. C'est d'ailleurs une des raisons de la création de l'interface « expert »[8], qui est à l'origine de ce troisième prototype. En effet, comme précisé dans le chapitre 4, ce système fonctionne sur base de nombreux concepts qui sont susceptibles de changer.

CosmicXpert ayant, entre autres, la responsabilité d'enseigner aux mesureurs la méthode de mesure COSMIC-FFP, il est nécessaire que celui-ci puisse rendre compte des modifications qui sont apportées à la théorie de la mesure elle-même. Il faut donc permettre aux experts de la mesure d'introduire ces changements dans la base de connaissances du système.

Néanmoins, la méthode COSMIC-FFP étant devenue une norme ISO [1], certains concepts (tels que les concepts topologiques) sont stables et ne devraient prêter que très rarement à modification.

6.1.2 Faiblesses des prototypes précédents

Au seuil du travail effectué en 2002-2003 par Gruselin et Vilz, CosmicXpert, de par son formalisme, est devenu une application plus facilement maintenable. En effet, le premier prototype était basé sur le format de fichier RTF et sur une base de données ACCESS pour

[8] Comprendre par interface « expert », l'interface de CosmicXpert destinée aux experts de la mesure COSMIC-FFP.

lier les fichiers entre eux. Seule une vérification manuelle de type « inspection » était alors possible.
L'inspection, définie dans le chapitre 2, permet de détecter les erreurs dans une base de connaissances par simple relecture. Ce qui laisse supposer qu'il peut toujours exister un grand nombre d'erreurs. De plus, l'inspection n'était que théoriquement réalisable, le nombre de fichiers la rendant extrêmement périlleuse en pratique.

Le deuxième prototype, développé par Gruselin et Vilz, ouvre les portes à une vérification automatique. Ce dernier est basé sur le format de fichier XML, comme nous venons de le voir dans le chapitre précédent.

Néanmoins, une vérification automatique présuppose une base de connaissances dans un état cohérent. Or, une des premières tâches que nous avons effectuée pendant le stage était l'implémentation d'un niveau supérieur d'indexage[9] pour faciliter la recherche par mot-clé dans l'interface « mesureur » de l'application.
Le développement de cet index a révélé une série d'anomalies dans la base de connaissances. Plus précisément, il s'est avéré que la base contenait de nombreuses déficiences. Pour rappel, une déficience se produit lorsqu'il existe des entrées valides vers une base de connaissances qui ne contient aucune règle relative à ces entrées. En effet, certains mots-clés étaient présents dans la base de connaissances mais ne référençaient aucun concept topologique. Ces erreurs ont dû être corrigées avant de pouvoir poursuivre. Dans la suite du développement de l'interface consacrée aux experts de la mesure COSMIC-FFP, d'autres erreurs se sont encore dévoilées. Il s'agissait principalement d'erreurs de cohérence.
Les techniques manuelles et semi-automatiques utilisées par Gruselin et Vilz pour vérifier les erreurs de la base de connaissances de CosmicXpert n'apportent qu'une réponse partielle au problème.
Comme l'ont fait remarquer Gruselin et Vilz dans [4] :

> *« Les techniques dites automatiques sont de loin les plus avantageuses. Avec ce type de technique, les ressources nécessaires (temps, personnes) se réduisent au minimum, mais la mise en œuvre de cette technique peut être assez complexe. Dès lors, si nous voulons un système de très haute qualité, il est nécessaire de prendre le temps*

[9] Le niveau d'indexage est une des nouvelles fonctionnalités introduites dans le troisième prototype de CosmicXpert. Celle-ci est présentée dans l'Annexe 4.

nécessaire à l'adaptation du système pour utiliser ces différentes techniques tout en ne changeant pas les fonctionnalités de base de notre système expert. »

C'est dans cette optique que nous avons développé l'interface permettant d'effectuer toutes les modifications de la base de connaissances.

6.2 Redéfinition de la problématique

Comme nous l'avons vu dans le second chapitre, les processus de vérification et validation sont des étapes incontournables dans le développement de système à base de connaissances. Nous avons orienté notre objectif vers la suppression de l'intermédiaire qu'est l'ingénieur de la connaissance au profit d'une interface intelligente permettant aux experts du domaine d'introduire, de modifier ou de supprimer eux-mêmes des connaissances dans le système. En plus de ces tâches, cette interface doit également assurer la vérification automatique de la base de connaissances. Par vérification automatique, nous entendons vérification au moment même où l'expert introduit une modification dans le système, par opposition à une vérification statique, qui ne s'exécute que sur un état donné de la base de connaissances. Dès que l'expert demande à effectuer un changement dans la base de connaissances, le système va lui imposer toute une série de contraintes, permettant d'assurer la vérification de la base. Nous présentons les modalités d'implémentation de ce mécanisme plus en détail dans le chapitre suivant.

6.3 Comparaison des trois prototypes

Le premier prototype implémentait toutes les fonctionnalités pour l'ensemble des acteurs du système, comme le montre la Figure 4.10. Trois interfaces particulières avaient été développées pour les différents acteurs du système, à savoir le mesureur, l'expert et l'administrateur.

Durant la réalisation du second prototype, Gruselin et Vilz, se sont concentrés sur les techniques de vérification et de validation de la base de connaissances du système, et sur l'interface mesureur du système.

En quelque sorte, le troisième prototype réalise la jonction entre le premier prototype, dans lequel toutes les fonctionnalités du mesureur, de l'expert et de l'administrateur avaient été implémentées, et le second prototype, basé sur la technologie XML permettant la vérification et la validation du système, dans un environnement distribué.

6.4 Description des fonctionnalités du troisième prototype

Le troisième prototype de CosmicXpert a, entre autres, comme objectif de compléter le second prototype, afin de permettre aux experts de réaliser des modifications sur la base de connaissances. Le second prototype se limitant à l'interface du mesureur, nous nous sommes attachés à développer une interface pour l'administrateur, mais surtout pour l'expert.

L'interface de l'administrateur n'est volontairement pas présentée ici, car elle n'apporte pas de signification au point de vue de l'automatisation de la vérification de la base de connaissances de CosmicXpert. Néanmoins, une description plus précise est donnée dans l'Annexe 4.

L'interface expert permet à tout expert de la mesure COSMIC-FFP, d'introduire un nouveau concept, de modifier ou de supprimer un concept (mot-clé, concept topologique, cas problème, thème, fait, recommandation, mot d'index) existant dans la base de connaissances de CosmicXpert.
La Figure 6.1 montre les différentes fonctionnalités offertes aux experts par l'interface. Elles sont représentées à travers le diagramme des cas d'utilisation (*Use Case*) du système. Ces cas d'utilisation sont détaillés dans l'Annexe 2.
Dans cette figure, les ovales sur fonds gris représentent un groupe de fonctionnalités associées à un concept particulier. Ils ne donnent pas nécessairement lieu à une matérialisation sous forme d'une interface. Les ovales sur fonds blancs, quant à eux, représentent chacun une fonctionnalité particulière, identifiée par un cas d'utilisation. Ils sont modélisés par une interface dans le système.

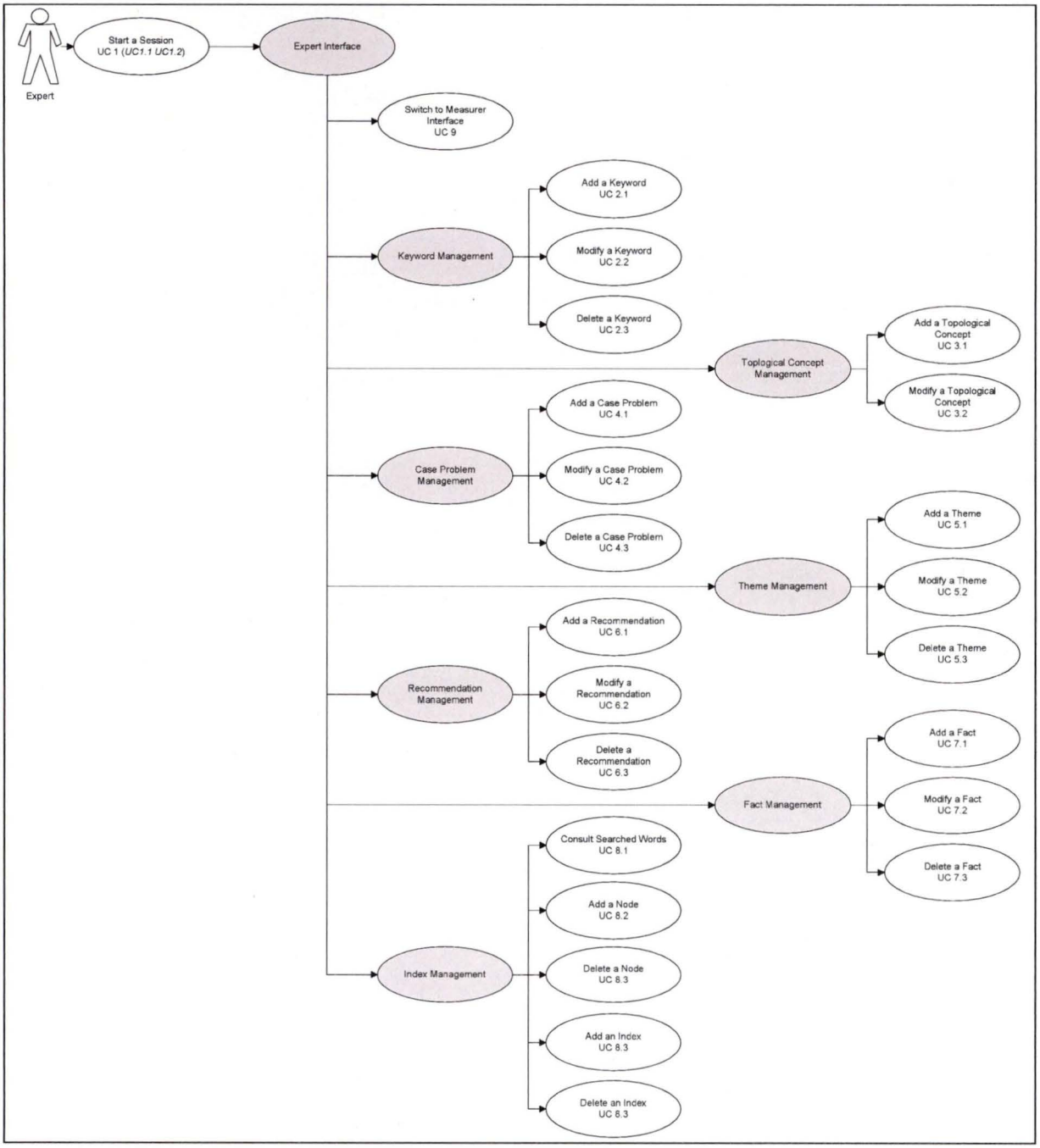


Figure 6.1 *Diagramme des cas d'utilisation de l'interface expert du troisième prototype de CosmicXpert*

Maintenant que nous avons fait état des différentes exigences émises pour le développement du troisième prototype de CosmicXpert, nous sommes en mesure de présenter les modalités d'implémentation de cette interface et plus particulièrement du mécanisme de vérification de la base de connaissances qui l'accompagne. Cette description est donnée dans le chapitre suivant.

Chapitre 7 : Développement de l'interface de l'expert

7.1 Méthodologie de développement

Avant d'entamer la description des particularités d'implémentation de notre système, nous allons présenter la méthodologie suivie durant l'ensemble du développement de l'application. Pour ce faire, nous nous référerons aux théories de méthodologie de développement logiciel exposées par Habra et Alexandre dans [26], ainsi que par Abrahamsson et al. dans [27].

De façon à pouvoir exposer notre méthodologie de développement, il est nécessaire de rappeler quelques concepts fondamentaux.

Le premier est celui de *processus logiciel*. Le processus logiciel est un « *ensemble structuré d'activités utilisées, de méthodes et de pratiques suivies pour la production et la maintenance du produit logiciel et des produits associés* »[26].

Le deuxième concept sur lequel il est nécessaire d'insister est celui de *cycle de vie du logiciel*. Il s'agit en réalité d'une « *ossature comprenant les processus, les activités et les tâches impliquées dans le développement, le fonctionnement et la maintenance d'un produit logiciel, couvrant la vie du système depuis la définition de ses exigences jusqu'à la fin de son utilisation* »[28].
Comme nous pouvons le constater, un cycle de vie est donc un agencement de processus et de pratiques couvrant l'ensemble de la vie du système.
Le processus de développement d'un logiciel est appelé cycle de vie parce qu'il décrit la vie du produit logiciel à travers son implémentation, sa mise en production, son utilisation et sa maintenance.

Un cycle de vie peut également être considéré comme un *outil de gestion de projet* permettant :

- d'identifier les différentes phases (processus et activités),
- d'identifier les délivrables,
- d'identifier les entrées/sorties entre les phases,
- d'ajouter une dimension temporelle au processus (début/fin),
- de distribuer les rôles et les responsabilités.

De façon générale, on peut dire qu'il est important pour toute organisation de choisir un bon cycle de vie. Ce choix est d'autant plus important qu'il va déterminer la qualité et le suivi de la qualité de la production logicielle.
Cependant, il faut savoir que, comme tout modèle, les cycles de vie constituent une vue idéalisée. Il sera donc impossible de choisir un cycle de vie parfaitement adapté à un contexte particulier. En revanche, il est possible de faire coexister plusieurs cycles de vie, au sein d'une organisation, ou même d'un projet.

Il est également important de préciser les définitions de *besoins*, de *caractéristiques* et d'*exigences logicielles*. Leffingwell et Widring [29] représentent ces différents concepts à travers la Figure 7.1.

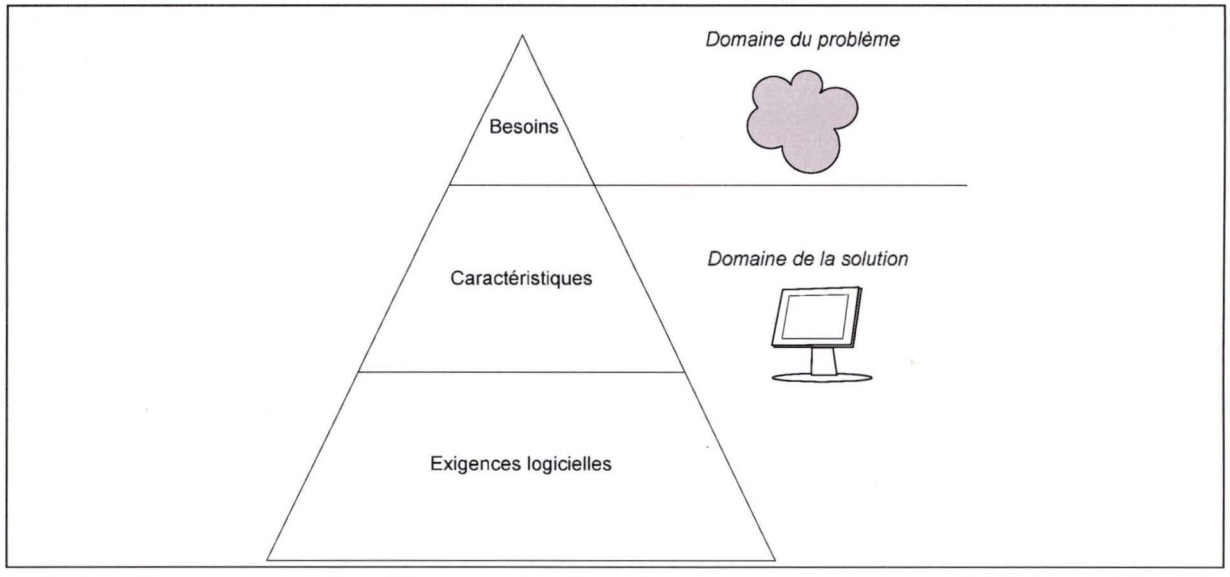


Figure 7.1 *Pyramide des besoins*

Toujours selon [29], les besoins (*needs*) sont exprimés par les utilisateurs du système et par toutes les personnes intéressées par le système. Les besoins représentent l'expression du problème à résoudre.
Les caractéristiques (*features*) correspondent aux services fournis par le système satisfaisant un ou plusieurs besoins des utilisateurs.
Une fois qu'on a établit l'ensemble des caractéristiques et que l'on a trouvé un accord avec le client, on peut discuter d'exigences (*requirements*) plus précises à imposer au système. Ces exigences plus précises sont les exigences logicielles. Si elles sont vérifiées par le système, on est sûr que les caractéristiques seront satisfaites.

En ce qui concerne notre développement, nous avons opté pour un cycle de vie basé sur le modèle par prototypage (*prototyping*). Ce modèle, comme le montre la Figure 7.2 issue de [26], a pour objectif d'être le plus réactif possible à l'utilisateur, en lui permettant de valider les spécifications par rapport à ses besoins réels, à des changements de caractéristiques du système ou à des modifications dans ces exigences. En effet, la validation d'une version des spécifications se fait par la démonstration, à l'utilisateur, du prototype associé à cette version.

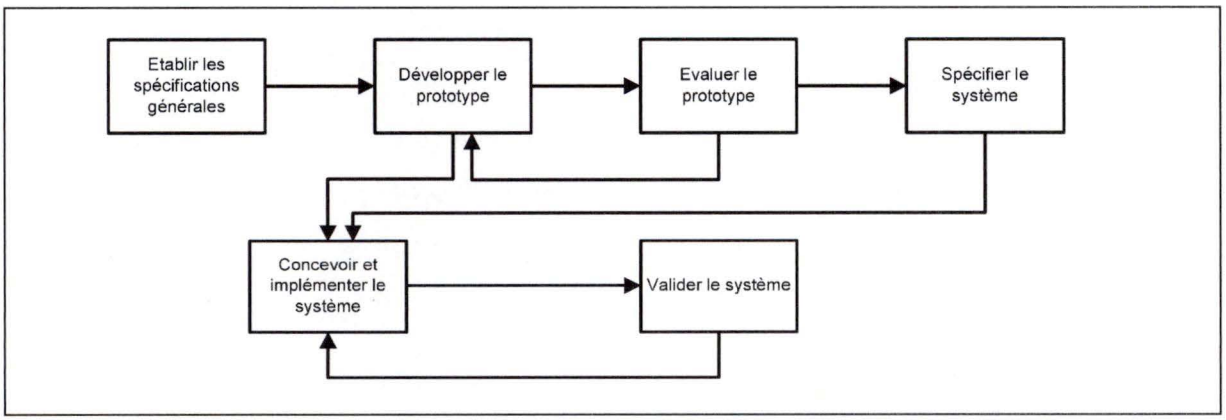


Figure 7.2 *Le modèle de cycle de vie par prototypage*

Ce cycle de développement est particulièrement adapté aux projets pour lesquels l'utilisateur définit seulement des objectifs généraux imprécis. Le développeur va l'aider à les préciser via des développements courts (prototypes), servant à identifier les exigences, par des suggestions.

Ce contexte de travail correspond particulièrement à notre projet, puisque les besoins étaient parfaitement définis, de par l'historique du logiciel, mais les caractéristiques étaient à

préciser. Nous connaissions les besoins de construire une interface permettant à un expert de modifier la base de connaissances de CosmicXpert, tout en assurant la vérification automatique, mais les caractéristiques et les exigences étaient assez vagues. Ils se sont précisés petit à petit, à travers toute une série de rencontres avec le commanditaire, Jean-Marc Desharnais.

De façon plus précise, nous pouvons dire que nous avons opté pour un cycle de vie par prototypage avec prototype évolutif (par opposition à *prototype à jeter*). Ceci signifie que l'approche utilisée est une approche par versions successives, en validant avec l'utilisateur ce qui a été fait. Si les exigences de l'utilisateur correspondent aux spécifications (au prototype), alors un nouvel aspect du système peut être pris en compte. Sinon, il faut corriger le tir.

Maintenant que nous avons décrit le cycle de vie utilisé, attardons nous un instant sur la méthodologie suivie.
Dans la même optique de réactivité face aux exigences de l'utilisateur, nous nous sommes tout naturellement tournés vers les méthodologies dites *Agile*. Ces méthodologies ont pour vocation d'être plus adaptatives et plus rapides en réponse à des processus de développement parfois trop lourds. Elles ont été développées par Larman, dans les années 90. Larman, cité dans [27] les définit comme suit :

> « *Les méthodes de développement Agile appliquent un développement itératif et évolutif limité dans le temps, un planning adaptatif, promeuvent des solutions évolutives, et incluent d'autres valeurs et pratiques encourageant l'agilité, la rapidité et une réponse flexible au changement.* »

Les méthodes Agile sont régies par la « *Software Manifesto* » [30] qui reprend les principes suivants :

- individus et interactions au lieu de processus et outils,
- travail du logiciel au lieu de documentation compréhensive,
- collaboration avec le client au lieu de négociation du contrat,
- réponse au changement au lieu de suivi du plan.

Dans le cadre du développement de l'interface de l'expert de CosmicXpert qui nous incombe, nous avons retenu les méthodes Agile telles que les *Crystal Families*.

Les Crystal Families ont été mises au point par Cockburn [31]. Il s'agit plus exactement d'un ensemble de méthodologies permettant de sélectionner la méthode la plus adaptée à chaque projet.

L'approche Crystal inclut des principes pour adapter les méthodologies en fonction du contexte, au regard de la taille et de la criticité du projet.

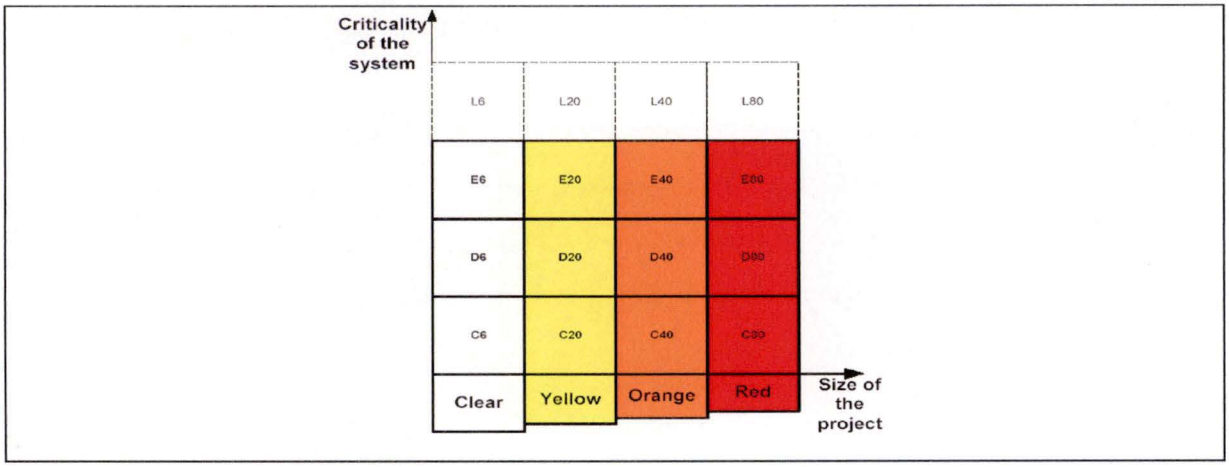


Figure 7.3 *Représentation du choix de la méthode Crsystal*

La Figure 7.3 indique le choix d'une méthodologie de la famille Crystal. Dans cette figure, il y a quatre caractères indiquant la criticité :

- *Comfort* (C), représente une perte de confort pour l'utilisateur en cas de crash du système.
- *Discretionary Money* (D), représente une perte minime d'argent en cas de crash du système.
- *Essential Money* (E), représente une perte importante d'argent en cas de crash du système.
- *Life* (L), représente une perte de vies humaines en cas de crash du système.

Il y a également quatre caractères indiquant la taille du projet. Ceux-ci sont :

- moins de 6 personnes,
- entre 6 et 20 personnes,
- entre 20 et 40 personnes,
- entre 40 et 80 personnes.

Le projet de développement lié à CosmicXpert peut donc être classé dans la première catégorie du tableau, c'est-à-dire, C6, puisqu'un crash du système entraînerait une perte de confort pour l'utilisateur et que l'équipe de développement ne compte que deux membres.
Cette classe est liée à la méthodologie appelée « *Crystal Clear* ». Cette méthodologie a été mise au point pour les tout petits projets, elle prévoit la livraison incrémentale de parties du logiciel, l'avancement est vérifié à partir des livraisons de logiciel plutôt que sur base de documents, l'utilisateur est directement impliqué à des fins de tests.

On voit donc apparaître une cohérence entre le cycle de vie choisit et la méthodologie suivie au sein de ce cycle de vie. L'objectif premier est d'éviter les processus lourds, les documents inutiles et de se concentrer sur le logiciel. De plus, la construction de l'interface pour l'expert étant un projet de développement court (4 mois), et de criticité faible, ce type de méthodologie et de cycle de vie sont particulièrement appropriés.

7.2 Démarche

7.2.1 Critères de vérification

Avant de pouvoir mettre en place un système de vérification pour notre système expert, il nous est nécessaire de dresser une liste des critères de vérification sur lesquels nous allons porter notre attention.
Nous sommes aidés dans cette tâche par [13], [14] et [15] pour établir le Tableau 2.1 , reprenant les principaux types d'erreurs rencontrés dans les systèmes à base de connaissances.

Le système de vérification que nous souhaitons mettre en place a la particularité suivante : il réalise une vérification en amont, c'est-à-dire, au moment même où l'expert demande la modification de la base de connaissances.
Pour cela, il est nécessaire de tenir compte d'un autre concept, hérité du monde des bases de données (cf. chapitre 3), le concept d' « *Intégrité* ». Pour rappel, il s'agit de maintenir la validité du système évoluant, et ce tout au long de sa vie. Nous expliquons ce point plus en détail dans la section suivante.

7.2.2 Processus de vérification

Cette section a pour but de montrer les moyens dont nous disposons pour réaliser notre objectif de vérification de l'intégrité de la base de connaissances de CosmicXpert.

7.2.2.1 Prise de contact avec la base de connaissances

Avant de pouvoir élaborer une technique de vérification, il est nécessaire d'étudier la base de connaissances de CosmicXpert plus en profondeur.
Cette étude est menée dans un premier temps de façon indirecte, à travers l'ajout de fonctionnalités annexes au système, telles une interface pour l'administrateur et un niveau supérieur d'indexage sur l'interface du mesureur.
L'interface de l'administrateur a pour objectif de permettre à l'administrateur de gérer l'ensemble des paramètres des utilisateurs du système, comme les noms d'utilisateur, les mots de passe, les dates d'expiration de ces mots de passe, et de façon générale, les droits d'accès.
Le niveau supérieur d'indexage, quant à lui, a pour but d'aider le mesureur dans sa recherche d'un mot-clé, en lui offrant la possibilité d'introduire un mot quelconque via une saisie libre.

7.2.2.2 Les fichiers XML

Dans un deuxième temps, nous nous plongeons plus en profondeur dans l'étude de la base de connaissances de CosmicXpert, en examinant la structure des fichiers qui la composent.
Depuis la réalisation du prototype 2, tel que mentionné dans le chapitre 5, la base de connaissances de CosmicXpert est constituée d'un ensemble de fichiers XML. Ces fichiers permettent une représentation formelle de la base et en facilite la vérification, grâce aux schémas XML contenant l'ensemble des contraintes imposées sur le contenu des fichiers XML. Ces contraintes représentent un ensemble de règles à vérifier pour garantir l'intégrité de la base de connaissances.

Nous présentons ici un exemple de ce type de contraintes, l'ensemble de toutes les contraintes de la base de connaissances est repris dans l'Annexe 1, à travers la charte de structuration des fichiers XML.
La Figure 7.4 montre une représentation générée par l'outil XMLSpy, du schéma XML structurant les documents de modélisation d'un Concept Topologique. Le fichier contenant cette structure est le fichier `tc.xsd`.

Nous allons utiliser ce schéma pour illustrer les contraintes d'intégrité imposées sur les documents représentant un Concept Topologique.

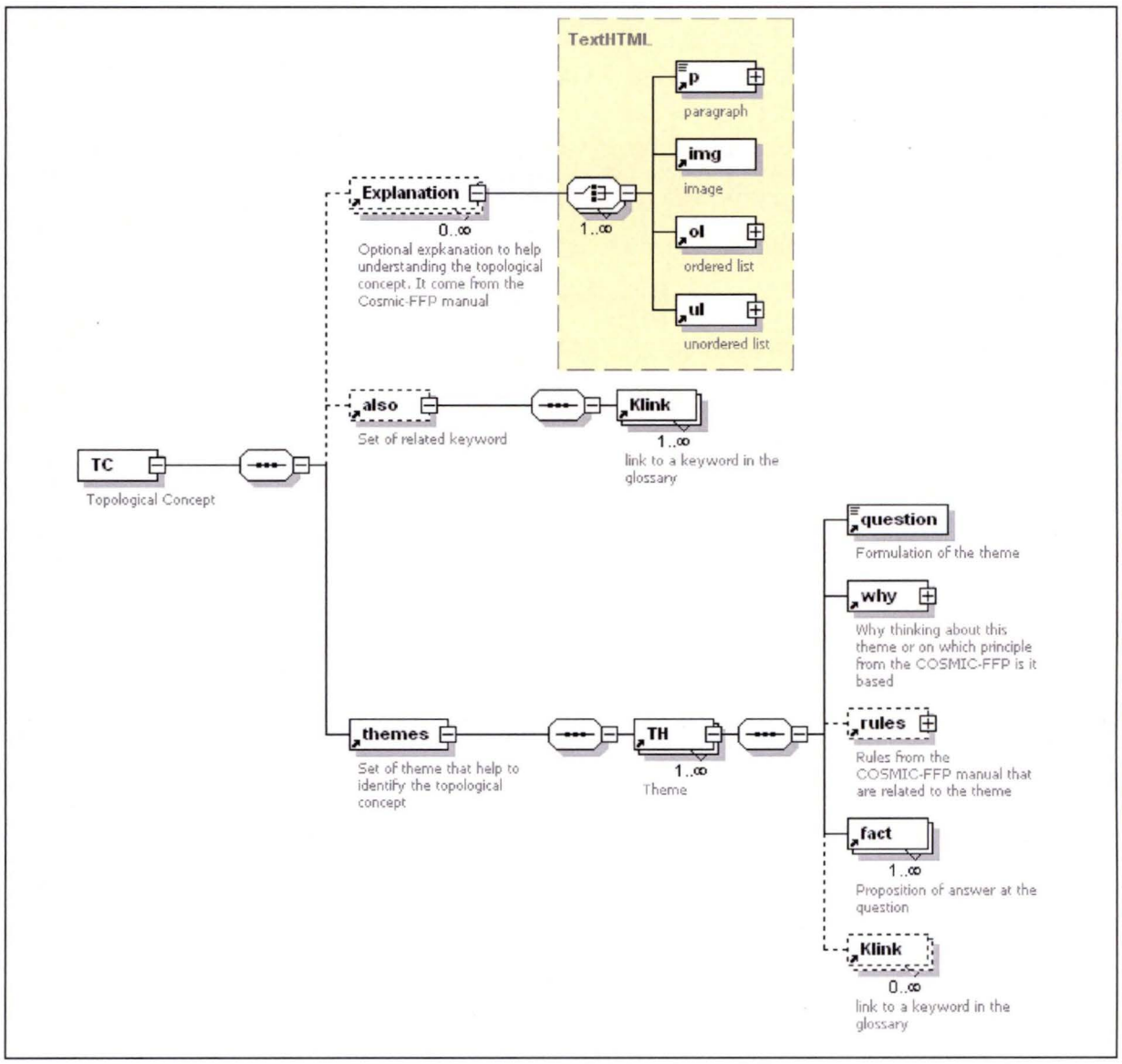


Figure 7.4 *Structure d'un Concept Topologique, dans le fichier* `tc.xsd`

Nous allons entamer la description de cette structure par l'explication de l'utilisation d'un type défini dans un document XSD annexe. Ce type de données a pour but de retrouver dans la vue créée par le XSL, les paragraphes, listes et images des documents RTF du premier prototype, en rendant possible la représentation de ces entités en XML. Pour cela, il a été défini d'utiliser un type de données appelé *TextHTML*. Nous utiliserons ce type de données dans la description de tous les types de documents.

Voici une représentation graphique du type TextHTML.

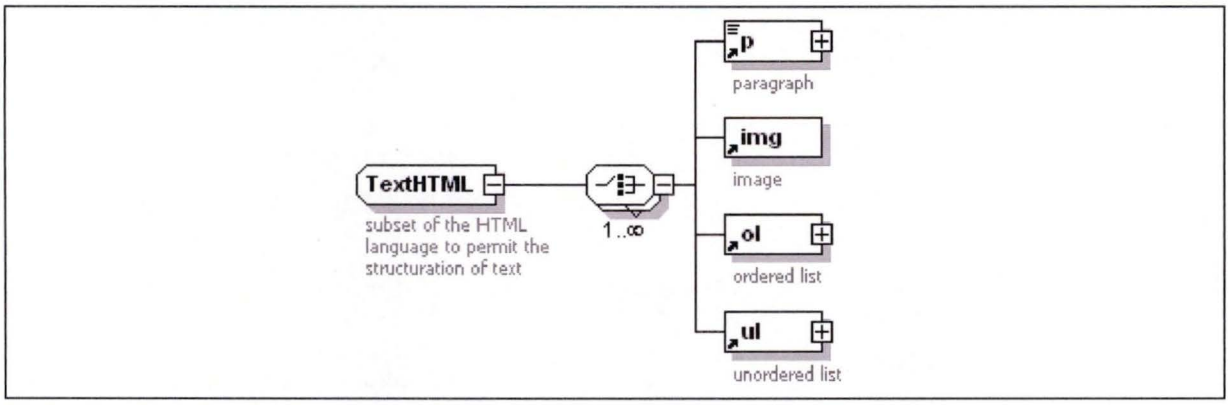


Figure 7.5 *Type TextHTML*

Chaque rectangle dans le schéma représente un élément XML pouvant être utilisé pour décrire une partie de document dans le langage. Chaque élément peut contenir des attributs et délimiter dans le fichier un type de données.

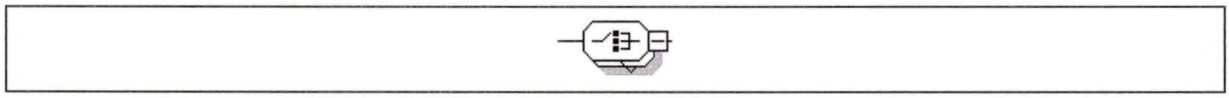


Figure 7.6 *Choix*

La Figure 7.6 représente un choix. Ce mécanisme, dans un schéma, signifie que l'élément ou le type à gauche peut contenir n'importe quel élément se trouvant à droite. Une cardinalité peut être associée à ce symbole et représente le nombre minimal et maximal d'éléments de droite contenus dans l'élément de gauche.

TextHTML est un type de données complexe car il peut contenir d'autres éléments ayant eux-mêmes un type de données.
Un élément de type TextHTML peut contenir un à plusieurs des éléments présentés dans le Tableau 7.1.

Tableau 7.1 *Description du type TextHTML*

Élément	Description	Type de données	Attributs	
p	Délimite un paragraphe	InsideP	align	Détermine la position horizontale du paragraphe (left, center, right)
img	Détermine l'emplacement d'une image à insérer	Vide	src	URL de l'image à insérer
			height	Hauteur de l'image
			width	Largeur de l'image
ol	Liste ordonnée			
ul	Liste non ordonnée			

InsideP est un type complexe dit mixte car il peut contenir du texte et des éléments mélangés. Ce type défini toutes les possibilité de représentation de texte à l'intérieur d'un paragraphe. Il est présenté plus en détail dans l'Annexe 1.

Nous poursuivons la description de la structure d'un document représentant un Concept Topologique, illustrée à la Figure 7.4. Le Tableau 7.2 détaille ce type de fichier, en décrivant chaque élément, son type de données, et ses attributs.

Tableau 7.2 *Description d'un document Concept Topologique*

Élément	Description	Type de données	Attributs	
TC	Délimite le document Concept Topologique		name	Nom du concept topologique
			defID	Identifiant du mot-clé définissant le concept topologique
Explanation	Délimite une éventuelle explication de concept topologique	TextHTML	name	Titre de l'explication

also	Délimite une série de mots-clés relatifs au concept topologique			
Klink	Représente un lien vers un mot-clé du glossaire	vide	idref	Identifiant du mot-clé lié dans le glossaire
Themes	Délimite les thèmes associés au concept topologique			
TH	Délimite un thème		id	Identifiant du thème
			cf	Pourcentage du lien entre le thème et son concept topologique
question	Délimite la formulation du thème en question	Text		
why	Délimite l'explication de la présence du thème	TextHTML		
rules	Délimite d'éventuelles règles apparentées au thème et provenant du manuel de mesure de Cosmic FFP	TextHTML		
fact	Délimite un fait	vide	name	Nom du fait
			CF	Facteur permettant le calcul pour déterminer la recommandation à proposer

Ce tableau illustre déjà toute une série de contraintes d'intégrité imposées par le schéma XML lié à ce type de document. Néanmoins, il en existe d'autres.

Tout d'abord, le schéma XML s'assure que les identifiants (`defID`, `id`) ont une valeur unique. De même, il vérifie que les clés étrangères (`idref`) sont bien formées, c'est-à-dire, respectent la forme de l'identifiant qu'elle référence. Cependant, les schémas XML ne permettent pas de vérifier qu'une clé étrangère référence effectivement un identifiant existant.

Pour poursuivre avec les attributs, le schéma XML vérifie le type de données des attributs. Ainsi, un champ du type entier (`cf`) ne pourra contenir qu'un entier et non une chaîne de caractères.

Notre analyse continue avec les éléments, au regard de la Figure 7.4. On peut observer que certains éléments (`Explanation`, `also` et dans un thème `rules`, `Klink`) sont facultatifs. D'autres doivent apparaître une et une seule fois (`themes`, et dans un thème `question`, `why`, `rules`). D'autres encore peuvent contenir une infinité d'instances (`Explanation`, `Klink`, `TH` et dans un thème `fact`, `Klink`).

Une dernière contrainte imposée par le schéma XML est que tous ces éléments doivent apparaître en séquence, comme illustré à la Figure 7.7. Ce mécanisme dans un schéma signifie que l'élément de gauche peut contenir une suite ordonnée d'éléments définis à droite du symbole.

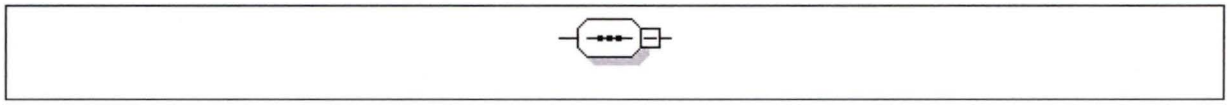


Figure 7.7 *Séquence*

7.2.3 Relevé des liens entre les concepts de la base de connaissances

Comme nous montre la section précédente, les fichiers XSD permettent d'imposer un certain nombre de contraintes sur les fichiers de la base de connaissances de CosmicXpert. Plus formellement, nous pouvons représenter l'un des types de contraintes : les liens qui doivent exister entre les fichiers de la base de connaissances, ou plus exactement, entre les connaissances de la base. Par exemple, le principe suivant est indispensable à la cohérence de la base de connaissances de CosmicXpert : « *tout mot-clé doit référencer au moins un concept topologique* ». Mais il en existe bien d'autres.
Nous avons choisi de représenter ces liens en les classant en fonction de l'opération qui est réalisée et du concept sur lequel s'applique cette opération.

Les tableaux qui suivent, doivent se lire comme suit :

> « Si un expert ajoute (*Add*), un mot-clé (K), alors, il faut qu'un lien physique (P) existe entre ce mot-clé et un concept topologique (TC) existant dans la base de connaissances.»

Les abréviations qui y sont reprises signifient :

- Index : représente un mot de l'index
- K : représente un mot-clé (*keyword*)
- TC : représente un concept topologique (*topological concept*)
- CS : représente une étude de cas (*case study*)
- CP : représente un cas problème (*case problem*)
- TH : représente un thème (*theme*)
- Fact : représente un fait (*fact*)
- Rec : représente une recommandation (*recommendation*)

Il existe deux types de liens entre les connaissances de la base :

- Les liens *physiques* (P) se manifestent par l'expression d'une contrainte dans un fichier XSD particulier de la base de connaissances. Le système impose à l'expert l'existence de ces liens entre deux connaissances.
- Les liens *logiques* (L) ne sont pas représentés par une contrainte particulière du système. Le système se contente d'avertir l'expert en cas de présence d'un lien logique entre deux connaissances, afin que celui-ci vérifie que les modifications qu'il a apportées à la base de connaissances correspondent bien à ses intentions.

Le Tableau 7.3 illustre les liens qui doivent exister entre deux concepts de la base de connaissances de CosmicXpert, lorsqu'un expert ajoute un concept particulier.

Tableau 7.3 *Représentation des liens qui doivent exister entre deux concepts lors d'un ajout*

Add	**Index**	**K**	**TC**	**CS**	**CP**	**TH**	**Fact**	**Rec**
Index	-	P						
K		-	P					
TC	P	P	-	P	P	P	P	P
CP	P	P	P	P	-			P
TH		P	P			-	P	L
Fact						P	-	L
Rec	P	P			P	L		-

Le Tableau 7.4 représente les liens qui doivent exister entre deux concepts de la base de connaissances de CosmicXpert, lorsqu'un expert modifie un concept particulier. Ce tableau est similaire au Tableau 7.3, puisque les opérations d'ajout et de modification d'un concept dans la base de connaissances ont des effets similaires.

Tableau 7.4 *Représentation des liens qui doivent exister entre deux concepts lors d'une modification*

Modify	**Index**	**K**	**TC**	**CS**	**CP**	**TH**	**Fact**	**Rec**
Index	-	P						
K		-	P					
TC	P	P	-	P	P	P	P	P
CP	P	P	P	P	-			P
TH		P	P			-	P	L
Fact						P	-	L
Rec	P	P			P	L		-

Le Tableau 7.5 représente les liens en cas de suppression d'un concept existant dans la base de connaissances de CosmicXpert. Ce tableau est différent des deux précédents car la suppression d'un concept entraîne des modifications différentes au sein de la base de connaissances. Par exemple, la suppression d'un mot-clé va entraîner des modifications de certains concepts topologiques, de certains cas problèmes, ainsi que des thèmes.

La ligne correspondant au concept topologique est vide, car le système n'autorise pas la suppression de ce concept, celui-ci étant stable dans la définition de la théorie COSMIC-FFP.

Tableau 7.5 *Représentation des liens qui doivent exister entre deux concepts lors d'une suppression*

Delete	**Index**	**K**	**TC**	**CS**	**CP**	**TH**	**Fact**	**Rec**
Index	-	P	P		P			P
K	P	-	P		P	P		P
TC	-	-	-	-	-	-	-	-
CP			P	P	-			P
TH			P			-	P	L
Fact			P			P	-	L
Rec					P	L		-

7.3 Utilisation des concepts issus des bases de données

7.3.1 Les transactions

Tel que précisé dans le chapitre 3, le mécanisme de transaction est particulièrement intéressant pour garantir l'intégrité d'une base de données. Ce mécanisme peut tout à fait être transposé et adapté de façon à réaliser la vérification de l'intégrité de la base de connaissances de CosmicXpert. C'est ce que la présente section tente d'illustrer.

Pour rappel, la définition d'une transaction telle qu'elle est utilisée par un SGBD est la suivante : « *la section d'un programme constituant une unité logique de traitement du point de vue des accès à la base de données* » [22]. Une transaction vérifie les quatre propriétés d'Atomicité, de Cohérence, d'Isolation et de Durabilité (A.C.I.D), également définies dans le chapitre 3.

Si un programmeur désire que les propriétés A.C.I.D. soient vérifiées pour une séquence d'instructions, il doit en faire une transaction. Celle-ci est effectuée dans l'univers des bases de données à l'aide de trois primitives offertes par le SGBD :

- Ouverture d'une transaction : `begin-transaction`
- Clôture d'une transaction avec confirmation : `commit-transaction`
- Clôture d'une transaction avec annulation : `abort-transaction`

Dans le cadre de CosmicXpert, l'expert désirant introduire, supprimer ou de manière générale, modifier des connaissances dans la base de connaissances, ne doit en aucun cas programmer de transaction. C'est le système lui-même, via l'interface, qui va gérer ce mécanisme[10].
Le langage Java, utilisé pour programmer le système de traitement de CosmicXpert, étant un langage de haut niveau, travaillant à l'aide d'une machine virtuelle, il est impossible de construire un outil gérant directement le mécanisme de transaction. En effet, la *Java Virtual Machine* (JVM) est trop abstraite que pour pouvoir réaliser la gestion des fichiers au niveau de l'architecture machine, or cette gestion des fichiers est nécessaire pour implémenter un mécanisme de transaction. Il nous est donc nécessaire de recourir à un subterfuge : l'utilisation d'un système de *log*.

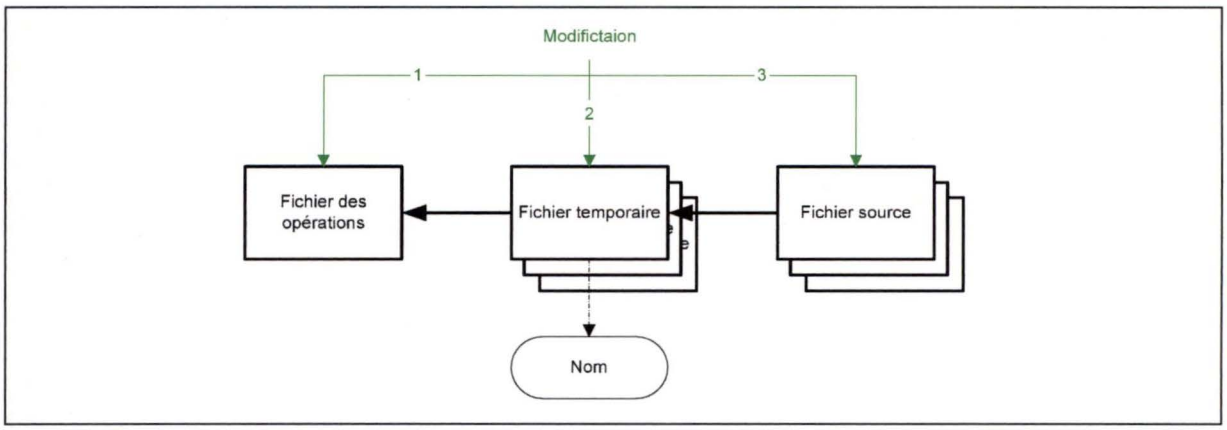


Figure 7.8 *Représentation du système de log*

La Figure 7.8 représente ce système de log. On peut constater qu'il est composé de trois ensembles de fichiers :

- Les *fichiers sources*, c'est-à-dire les fichiers XML qui vont faire l'objet d'une transformation suite à la modification d'une connaissance de CosmicXpert.
- Les *fichiers temporaires*, sur lesquels les modifications seront appliquées dans un premier temps. Le nom de ces fichiers temporaires est identique au nom du fichier « source » dont ils sont la copie, en ce compris le chemin

[10] Ceci ne constitue pas une différence par rapport aux SGBD, puisque des systèmes tels que Oracle, Access,... permettent également à un utilisateur, via leur interface, de réaliser des modifications sur les bases de données, sans que celui-ci ne doive programmer de transaction. Néanmoins, nous insistons sur le fait que notre système offre une certaine facilité d'utilisation.

d'accès de ce dernier au sein de la base de connaissances. Pour des raisons techniques le nom des fichiers temporaires contient également un numéro identifiant la transaction.

- Le *fichier des opérations*, qui reprend le type de l'opération à effectuer sur le fichier source (*Add*, *Modify*, *Delete*) et l'état de cette opération (*operation done* si l'opération a été complétée avec succès sur tous les fichiers temporaires, sinon le fichier reste vide).

En référence à la Figure 7.8, nous pouvons constater comment ces fichiers interviennent dans le mécanisme que nous avons développé :

- Lors d'une modification de la base de connaissances de CosmicXpert, le système écrit tout d'abord le type d'opérations (*Add*, *Modify*, *Delete*) à effectuer, sur les fichiers sources, dans le fichier des opérations (1).
- Les fichiers sources sont ensuite recopiés (2). Ces copies (c'est-à-dire les fichiers temporaires), une fois renommées, sont utilisées pour le reste des opérations.
- Une fois que les fichiers temporaires sont correctement mis à jour en fonction des modifications demandées, le fichier des opérations a été modifié de façon à contenir la mention « *operation done* ». Le contenu des fichiers temporaires est ensuite recopié dans les fichiers sources (3), parallèlement ils sont déplacés dans le dossier des *backups*, de façon à constituer une sauvegarde de récupération en cas de perte de la base de connaissances (nous reviendrons sur ce point dans la section 7.3.2).

De façon schématique, la Figure 7.9 représente notre mécanisme de transaction et sa réaction suite à un incident. Si un incident survient pendant l'une trois premières étapes, empêchant la suite de la transaction, aucune modification ne sera appliquée à la base de connaissances. Le système réalisera un *undo* de façon à effacer toute trace de la transaction. Si par contre, la mise à jour de tous les fichiers temporaires a pu être réalisée, alors on peut affirmer que la modification demandée par l'expert sera introduite dans la base de connaissances. En effet, le système va alors recopier le contenu des fichiers temporaires dans les fichiers sources. Si un incident survient durant cette étape, le système reprendra l'écriture, de façon à terminer la transaction. On parle dans ce cas d'un *redo*.

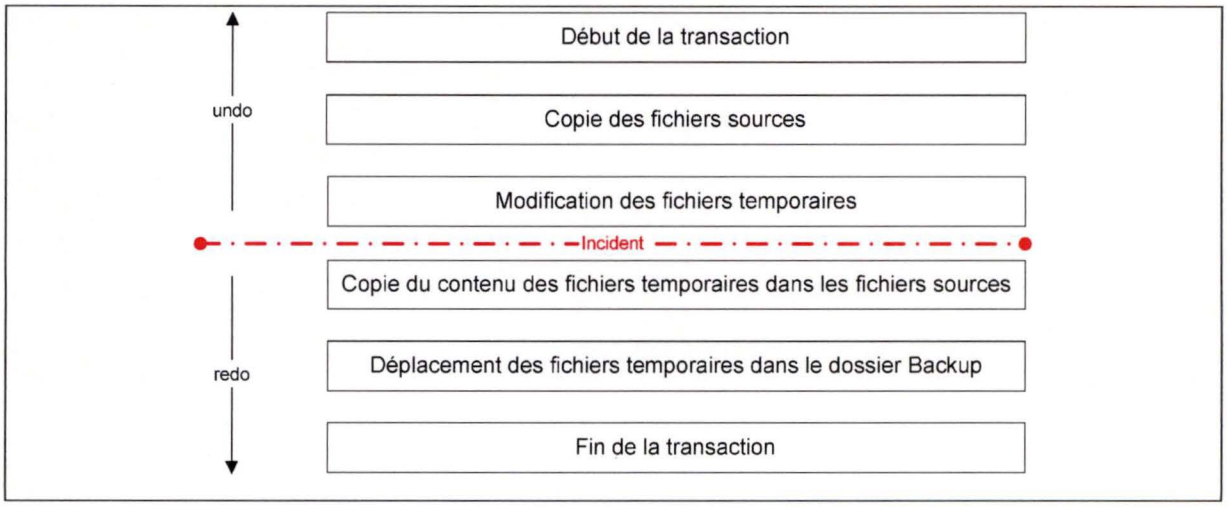


Figure 7.9 *Représentation du mécanisme de transaction de CosmicXpert*

Illustrons ce mécanisme par un exemple simple (un exemple plus complet est présenté au chapitre 8). Soit la modification de la définition d'un *mot-clé* et de l'une de ses relations avec un *concept topologique*, comme le montre la Figure 7.10. Dans ce cas, les fichiers sources concernés sont `glossary.xml`, contenant tous les éléments de description propres au mot-clé, et `xpert.xml`, contenant en particulier les associations entre un mot-clé et les concepts topologiques.

La transaction débute au moment où l'expert clique sur le bouton « *modify* », et de ce fait, confirme qu'il souhaite que le système prenne en compte les modifications qu'il vient d'apporter à un mot-clé. Le système va alors créer un fichier des opérations pour la transaction courante. Il va ensuite recopier les deux fichiers sources concernés, à savoir `glossary.xml` et `xpert.xml`. Il poursuivra la transaction en appliquant les mises à jour demandées sur les fichiers temporaires. Si un incident est survenu pendant l'une des étapes précédentes, le système réalisera un *undo* de façon à effacer tous les fichiers temporaires ainsi que le fichier des opérations. Il n'y aura donc plus aucune trace de la transaction. Sinon, le système pourra continuer la transaction. Il va alors recopier le contenu des fichiers temporaires dans les fichiers sources. Si un incident survient durant cette étape, le système effectuera un *redo* de façon à terminer la transaction.

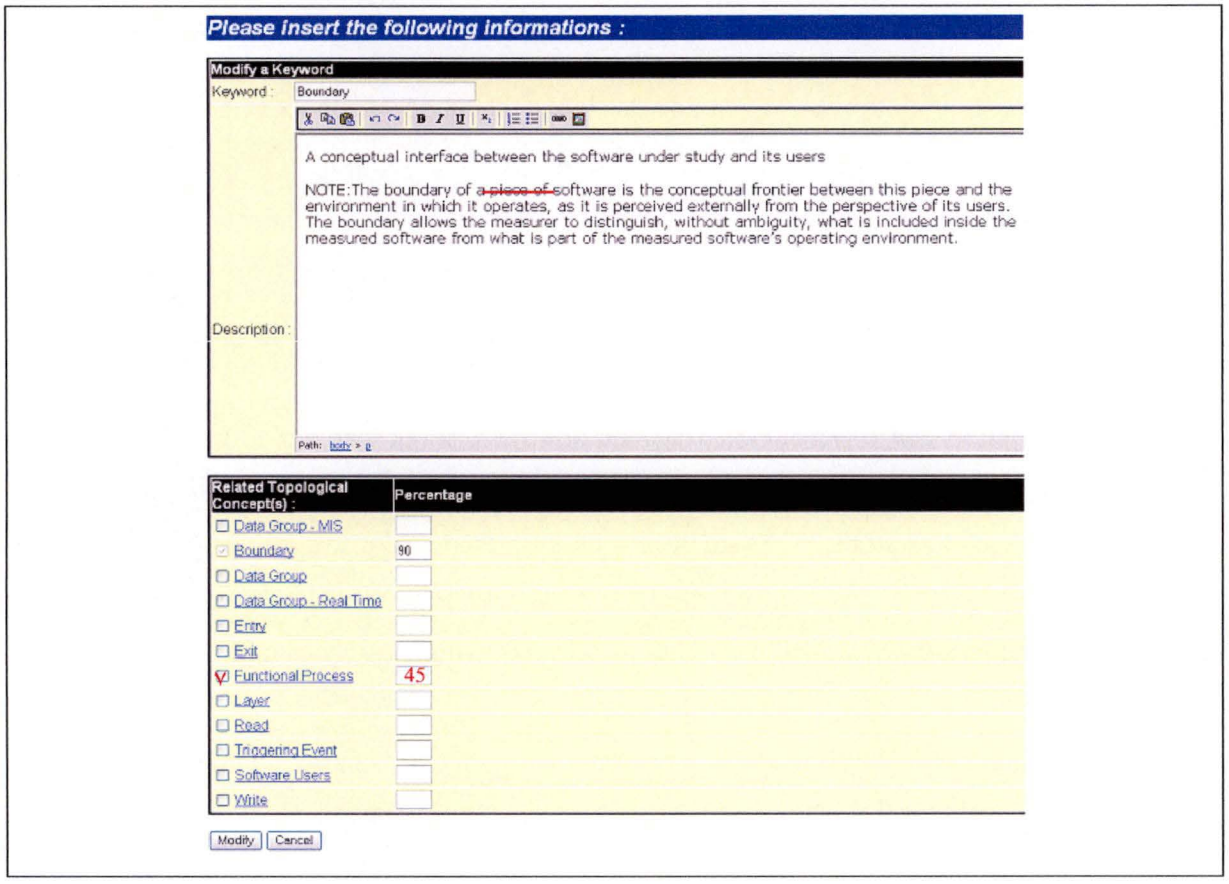


Figure 7.10 *Exemple de modification d'un keyword*

7.3.1.1 Transactions hiérarchiques

Comme nous l'avons vu dans le chapitre consacré aux bases de données, dans un SGBD, lors de l'exécution d'un programme, les transactions se déroulent séquentiellement, mais jamais en parallèle. Il n'est donc pas possible de définir une transaction comme étant formée de sous-transactions plus élémentaires.

Il en va de même avec le système de modification de la base de connaissances de CosmicXpert. Ceci ne constitue en rien une limitation car à toute opération est volontairement associée une et une seule transaction. Il s'agit bien d'une volonté de notre part, afin de réduire l'ensemble des modifications à apporter à la base de connaissances à quelques opérations élémentaires de copie de fichiers, de façon à minimiser le temps durant lequel les verrous sont posés sur les fichiers concernés. Nous reviendrons sur ce point dans la section 7.3.3.

7.3.1.2 Transactions et intégrité

Un problème existe toujours dans le monde des bases de données, concernant l'intégrité. Plus précisément, si la base de données est cohérente à l'entrée d'une transaction, alors, au sortir de la transaction,

- soit la base est encore dans cet état (abort, en cas d'incident),
- soit elle se trouve dans un nouvel état (commit) qui respecte les contraintes d'intégrité existantes.

Mais il est possible que certaines de ces contraintes ne soient satisfaites qu'à la fin de la transaction. Il est de ce fait possible que des états intermédiaires de la base de données ne respectent pas ces contraintes. Une solution à ce problème, proposée par les SGBD, est de postposer la vérification des contraintes, lors d'une transaction. Mais là encore, c'est au programmeur de spécifier les vérifications qui doivent être remises à plus tard.

Dans notre système, l'utilisation de fichiers temporaires retarde l'apparition d'états intermédiaires incohérents de la base de connaissances. Ce n'est qu'au moment où le contenu des fichiers temporaires est recopié dans les fichiers sources que les contraintes d'intégrité ne sont momentanément plus vérifiées. Lorsque la transaction se termine, les contraintes d'intégrité sont à nouveau respectées. Ces états intermédiaires sont sans conséquence pour l'intégrité de la base de connaissances, car la vérification des contraintes d'intégrité existant dans la base de connaissances n'est effectuée qu'au moment de l'utilisation du fichier XML auquel sont attachées ces contraintes. Les seules conséquences que pourraient apporter ces états intermédiaires incohérents, seraient que si un *mesureur* décide de consulter une connaissance dont la mise à jour est en cours, il ne puisse pas réaliser cette opération, une erreur l'en empêchant momentanément.

7.3.2 Protection contre les incidents

Le mécanisme de transaction exposé dans la section précédente, est une réponse à la garantie des propriétés A.C.I.D. Il est nécessaire cependant de disposer d'autres mécanismes permettant de réagir face à des incidents majeurs, survenant à l'intérieur ou en dehors d'une transaction. Le système doit, par exemple, être capable de gérer la destruction d'une partie ou de la totalité de la base de connaissances. Les propriétés visées ici, sont l'Atomicité et la Durabilité.

Ces deux problèmes sont définis, dans le monde des bases de données, comme suit :

- La propriété d'Atomicité précise qu'en cas d'incident au sein de la transaction, la base de données doit être remise dans l'état du début de la transaction. Si des modifications ont déjà été réalisées, elles doivent être défaites (*rollback*).
- La propriété de Durabilité, quant à elle, demande qu'en cas d'incident après la clôture de la transaction, les données, même corrompues, doivent être remises au moins dans l'état où elles étaient au sortir de la transaction.

Cette double protection est assurée de façon générale dans l'univers des bases de données, par la disponibilité de deux fichiers annexes : la sauvegarde (*backup*) et le journal (*log*). Ces mécanismes ont été exposés dans le chapitre 3.

Dans le cas de CosmicXpert, il est également possible d'assurer ces deux propriétés. Le mécanisme utilisé est assez similaire à celui des SGBD, puisqu'il fait appel à une *sauvegarde* et à des *fichiers de backup*.

La sauvegarde est une copie totale de la base de connaissances à un instant donné, réalisée manuellement par l'administrateur du système.

Les fichiers de backup, quant à eux, sont créés lors de l'exécution de toute transaction. Dès qu'une modification est demandée par un expert, le système va réaliser la mise à jour des connaissances via le mécanisme de transaction présenté plus haut. La particularité de ce mécanisme est de travailler sur des fichiers temporaires dont le contenu sera recopié dans les fichiers sources. Ces fichiers temporaires ne seront pas supprimés au sortir de la transaction mais déplacés dans un dossier *backup*. Ils constituent donc une version de récupération, comparable au *Journal de images après* présenté à la section 3.3.

Il est important de noter que si un fichier subit plusieurs modifications sur une certaine période, seule la version la plus récente de ce fichier sera conservée dans le dossier de backup.

7.3.2.1 Reprise suite à un incident

Le système de récupération de CosmicXpert que nous avons choisit d'implémenter est un système dit de reprise *à froi*d , tel que défini dans le chapitre consacré aux bases de données. Il permet de solutionner tous les incidents graves pouvant survenir à la base de connaissances, qu'il s'agisse d'une détérioration partielle ou même d'une perte totale de la base de connaissances.

Ce système se caractérise par deux étapes :

- Récupérer la dernière sauvegarde,
- Réappliquer les fichiers de backup enregistrés depuis la prise de cette sauvegarde. Puisque le dossier de backup ne contient que la dernière version de chaque fichier, il n'y aura pas d'écriture inutile, ce qui constitue une sérieuse optimisation.

La reprise dite *à chaud*, également définie dans le chapitre consacré aux bases de données, plus légère, n'a pas été implémentée en tant que telle mais est résolue à travers le mécanisme de transaction utilisé.

L'interface développée permet à l'administrateur du système de réaliser une copie de sauvegarde quand il le désire, sur le support de son choix. De même, si la base de connaissances n'est que partiellement détériorée, l'administrateur a la possibilité d'effectuer la récupération de manière automatisée.
Si la base de connaissances est totalement détruite, ou si les fichiers nécessaires au lancement de l'application ont été détruits, la récupération devra se faire manuellement.

7.3.3 Régulation de la concurrence

Toujours dans un souci de comparaison avec les principes issus des bases de données, nous nous attardons dans cette section sur la garantie de la propriété d'Isolation qui précise que chaque transaction doit se comporter comme si elle était seule à travailler sur la base de connaissances.

Dans le monde des bases de données, plusieurs classes de techniques existent pour éviter les interférences nuisibles. Néanmoins, la technique la plus économique et la plus souvent utilisée par les SGBD est la technique des protocoles restrictifs, dont l'exemple type est le principe de *verrouillage*. Ce mécanisme a déjà été défini dans le chapitre 3.

Le système que nous avons développé pour CosmicXpert s'appuie également sur ce principe de verrouillage. Toute modification au sens large (c'est-à-dire ajout, modification ou suppression), d'une connaissance nécessite la pose d'un *verrou exclusif* sur le fichier correspondant à cette connaissance. La pose d'un verrou, tout comme pour les bases de

données, est demandée implicitement par le système de transaction et est invisible pour l'utilisateur.

La mise au point de ce mécanisme de transaction s'est effectuée via plusieurs solutions, présentées dans les sections suivantes.

7.3.3.1 Solution 1

Dans la première solution, la pose des verrous sur les fichiers est effectuée dès le début de la requête de modification de la base de connaissances, c'est-à-dire, dès le moment où un expert décide d'ajouter, de modifier ou de supprimer une connaissance dans le système. La relâche des verrous peut être effectuée dès que les fichiers sources ont été mis à jour. Ce mécanisme est illustré à la Figure 7.11.
Les cadres sur fonds gris représentent des opérations nécessitant l'intervention de l'expert. Les cadres sur fonds blancs représentent des opérations réalisées de façon autonome par le système.

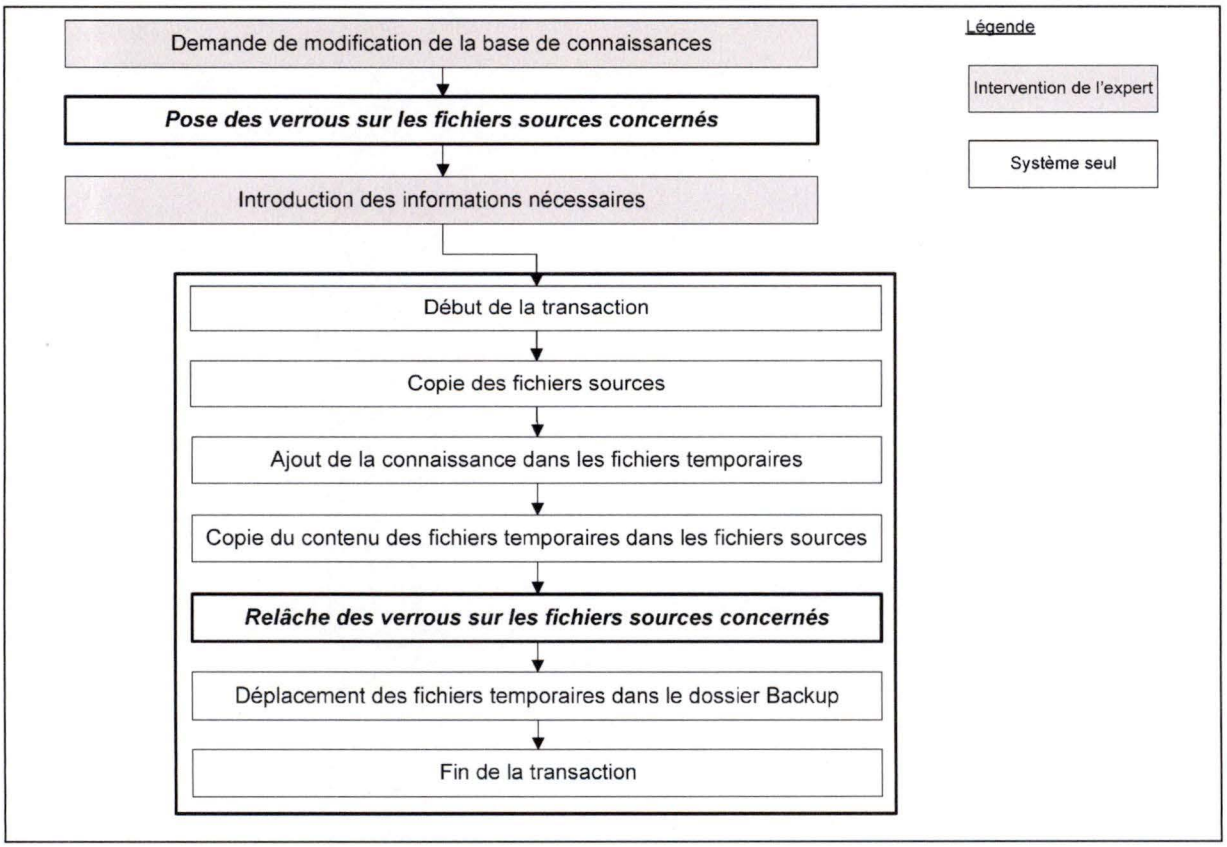


Figure 7.11 *Représentation de la première solution de verrouillage*

Ce principe à l'avantage d'être très simple, que ce soit au point de vue conceptuel ou au point de vue de l'implémentation. Mais ses limitations sont tout aussi évidentes. En effet, si les verrous sont posés sur les fichiers dès le début de la requête, tous ces fichiers seront inaccessibles aux autres experts pendant une longue période. De plus, ce laps de temps est indéterminé, puisqu'il dépend du temps nécessaire à l'expert pour introduire les modifications qu'il souhaite apporter à la base de connaissances. Or, cet expert peut être inactif à certains moments, retardant encore la libération des fichiers. Ce problème est communément appelé « *problème de la pause café* ». Malgré que le temps d'attente ne puisse être indéfini, grâce à la technologie utilisée (Java Server Page, serveur Tomcat), qui limite le temps d'inactivité d'un utilisateur, il va sans dire que cette solution est inacceptable de façon générale dans tout système distribué.

7.3.3.2 Solution 2

L'idée qui résulte des inconvénients de la première solution est de réaliser la pose des verrous après la saisie des informations, c'est-à-dire au début de la transaction.
La représentation de cette deuxième idée de solution est présentée à la Figure 7.12.

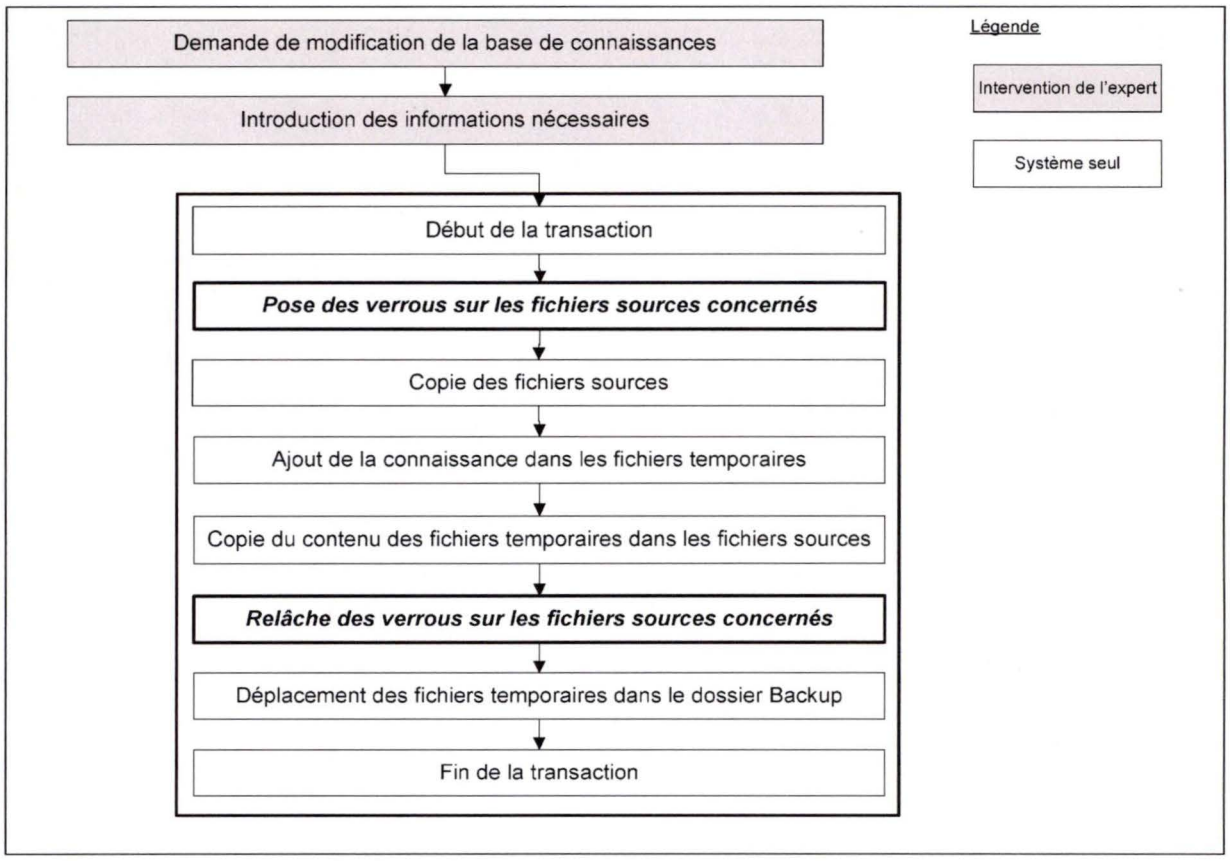


Figure 7.12 *Représentation de la deuxième solution de verrouillage*

Cette solution est plus efficace que la première, en terme de temps de pose des verrous. De plus, le « *problème de la pause café* » est résolu, puisque le laps de temps pendant lequel les verrous sont maintenus ne dépend plus de l'intervention de l'expert.
Cependant, cette solution souffre encore d'un défaut majeur, illustré à la Figure 7.13.
Deux problèmes sont envisageables à partir de cette figure.
Le premier est que, comme l'expert n°1 a déjà demandé la modification d'un keyword (fichier `glossary.xml`), l'expert n°2, qui souhaite également réaliser une modification sur l'un des mot-clé (identique ou non à celui de l'expert 1), ne sera prévenu de l'impossibilité de poursuivre son opération qu'après avoir saisi toutes les informations demandées par le système. Or, l'introduction de ces informations peut représenter une charge de travail non négligeable.
La raison pour laquelle il lui est impossible de poursuivre est que le système est incapable de poser un verrou sur le fichier, puisqu'un verrou de ce type existe déjà, de par la modification demandée par l'expert 1.

Une solution possible à ce problème, de façon à éviter que l'expert 2 ne doive réintroduire toutes les informations, serait de postposer la suite de la seconde transaction. Mais dans ce cas, la mise à jour résultant de cette opération conduirait à écraser la mise à jour effectuée par l'expert 1, puisque le fichier `glossary.xml` a été modifié depuis le début de la requête de l'expert 2. On parle dans ce cas de mise à jour perdue (*lost update*). Ce qui constitue notre deuxième problème.

Pour résoudre ce problème, il faudrait vérifier que le fichier `glossary.xml` n'a pas été modifié par un tiers, avant de poser le verrou. Mais cette solution ne résout que le cas où la connaissance modifiée par les deux experts est différente. De plus, sa mise en œuvre nécessite l'implémentation de toute une série de mécanismes de vérification complexes, pouvant diminuer les performances du système. On lui préférera donc un autre type de verrouillage.

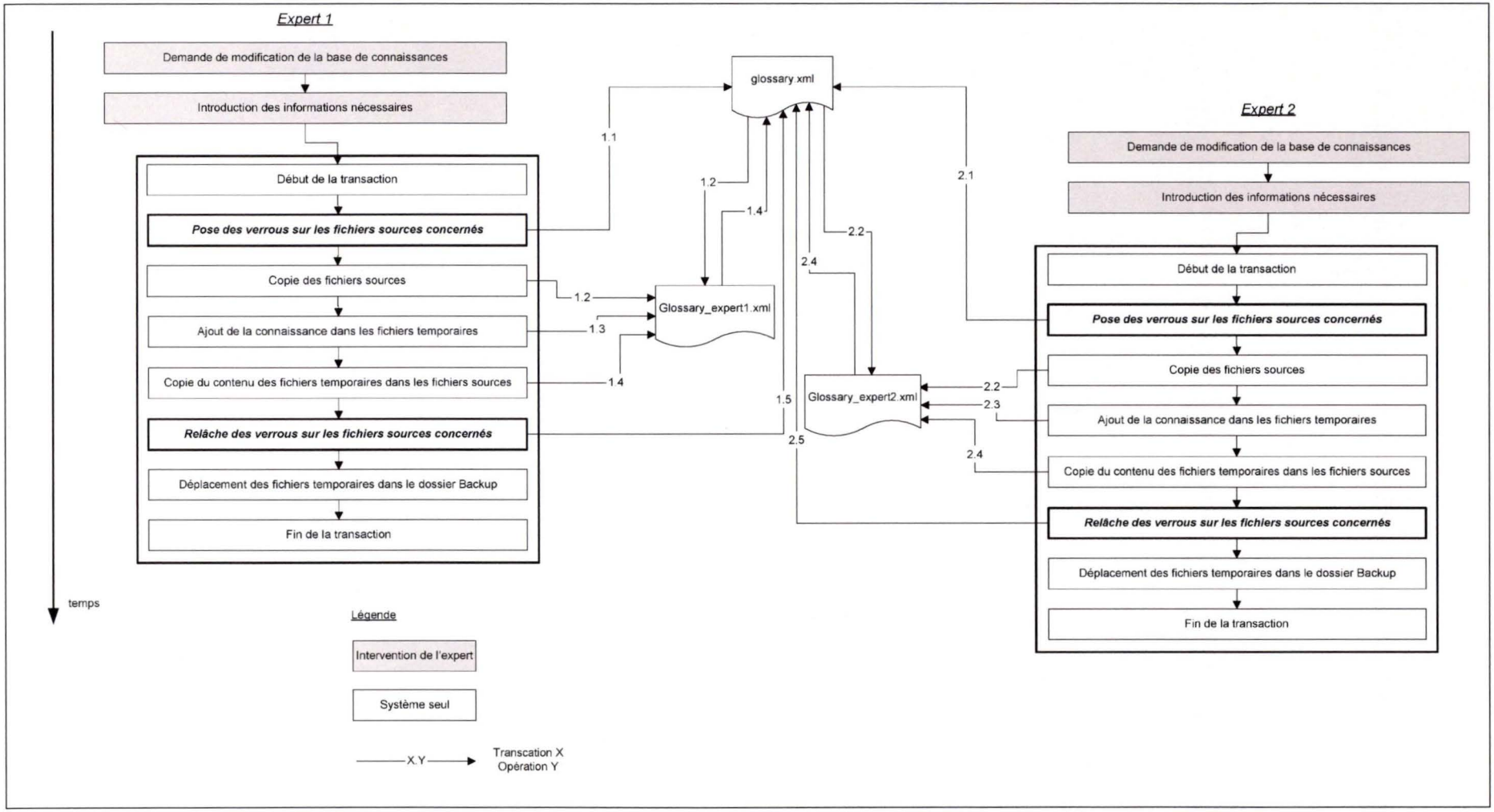


Figure 7.13 *Représentation du problème de la mise à jour perdue*

7.3.3.3 Solution 3

La troisième et dernière solution est basée sur un double mécanisme de verrouillage. Dans un premier temps, au début de la requête de modification d'un expert, un verrou est posé non plus sur la totalité du fichier mais bien sur la connaissance visée. De cette façon, on est certain que cette connaissance ne pourra pas être modifiée par un autre expert, et ce durant tout le processus. Ensuite, au début de l'exécution de la transaction, un autre verrou est posé au niveau du fichier contenant cette connaissance. De ce fait, on s'assure qu'un expert ne pourra pas être interrompu au milieu de son travail et qu'il n'y aura pas de mises à jour perdues.
Ce double mécanisme de verrouillage est présenté à la Figure 7.14.

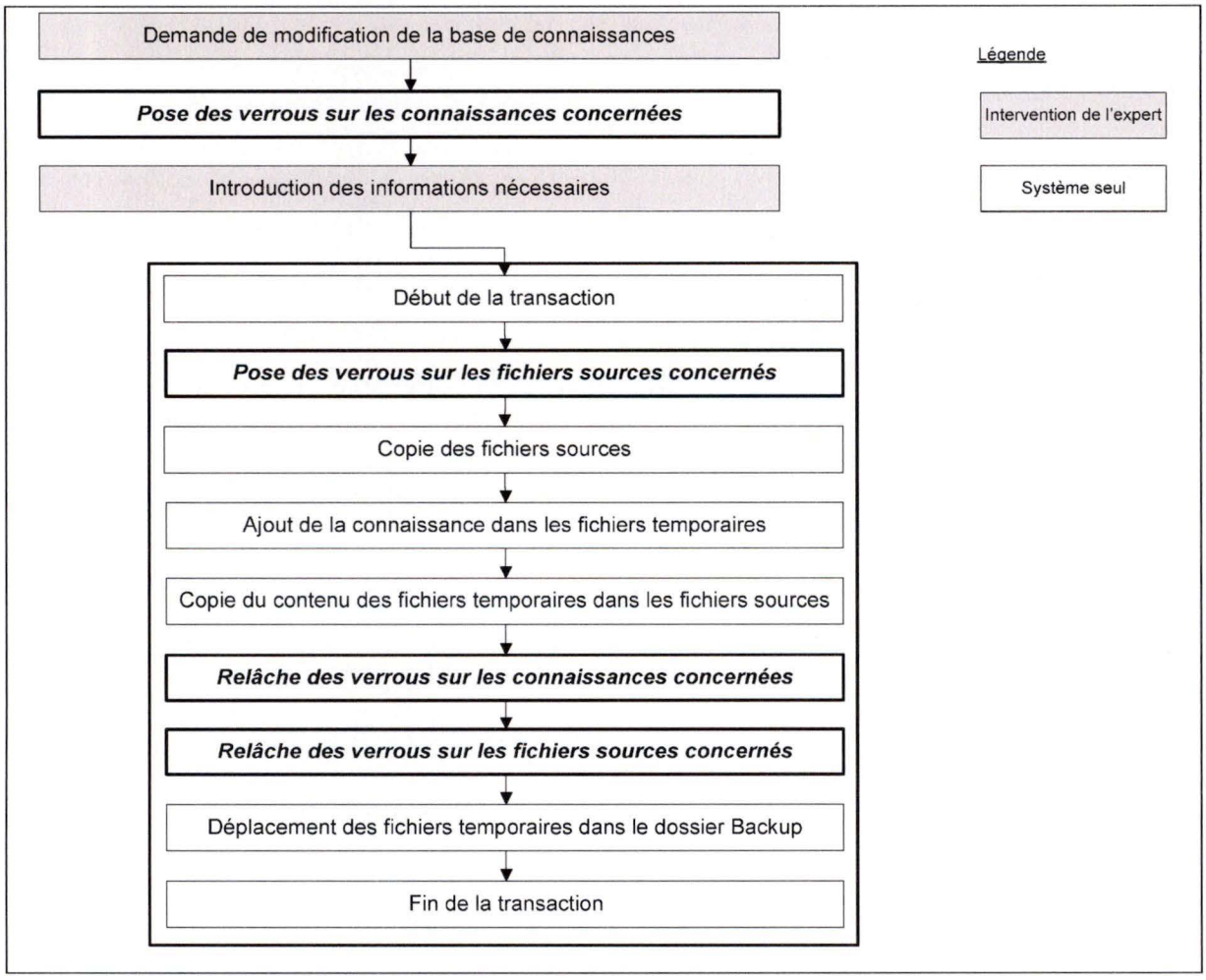


Figure 7.14 *Représentation de la troisième solution de verrouillage*

Cette solution à l'avantage de permettre le contrôle de la concurrence et, de ce fait, de garantir la propriété d'Isolation, tout en favorisant au maximum le parallélisme.

7.4 Utilisation de concepts propres à CosmicXpert

Au dessus de ces concepts tirés des bases de données viennent se greffer toute une série d'autres vérifications introduites par l'interface elle-même.
Ces vérifications ayant lieu dès l'introduction des données par l'utilisateur, un feedback rapide et adapté peut être apporté à l'utilisateur. D'autre part, le travail de vérification de la base de connaissances est simplifié, puisque seules les données conformes seront acceptées. Il est à noter toutefois que nous sommes ici à un niveau relativement élevé de vérification de la conformité des informations introduites.
La vérification effectuée en amont par l'interface peut être décomposée en deux niveaux distincts.

7.4.1 Les formulaires

Lorsqu'un expert désire apporter des changements dans la base de connaissances de CosmicXpert, par l'ajout, la modification ou la suppression d'un concept, il doit le faire via l'interface « expert ». Plus particulièrement, lorsque le système lui demande d'introduire des informations, il doit le faire via le remplissage des champs d'un *formulaire*.
Un formulaire, au sens d'un formulaire électronique, peut être défini comme étant un moyen rapide et aisé d'échange d'informations entre l'utilisateur d'un site Web et le serveur Web. De façon générale, ce formulaire se présente sous une forme comparable à son homologue papier.

La Figure 7.15 présente un exemple de formulaire tiré de l'interface de CosmicXpert. Il s'agit de la première partie des informations qu'un expert doit introduire lorsqu'il désire ajouter un cas problème.

L'usage de formulaires permet de réaliser une première vérification des informations, puisque l'expert ne sera pas en mesure d'introduire des informations non conformes aux champs du formulaire. Par exemple, si l'expert introduit une *chaîne de caractères* alors que le formulaire attend un *entier*, le formulaire ne pourra pas être validé et l'expert sera prié de réintroduire des informations valides.
Nous avons vu précédemment que les fichiers XML qui composent la base de connaissances de CosmicXpert sont contraints par les fichiers XSD qui leurs sont associés. On peut aisément constater que l'utilisation de formulaires permet de satisfaire à priori une partie de ces

contraintes, puisqu'ils permettent, entre autres, de contrôler les types de données, de forcer l'expert à compléter les champs obligatoires, l'utilisation de champs particuliers va imposer à l'expert d'introduire les informations en respectant les contraintes de séquences ou de choix des fichiers XSD (le champ « *Problem identification* » doit respecter la srtucture TextHTML définie en 7.2.2.2), …

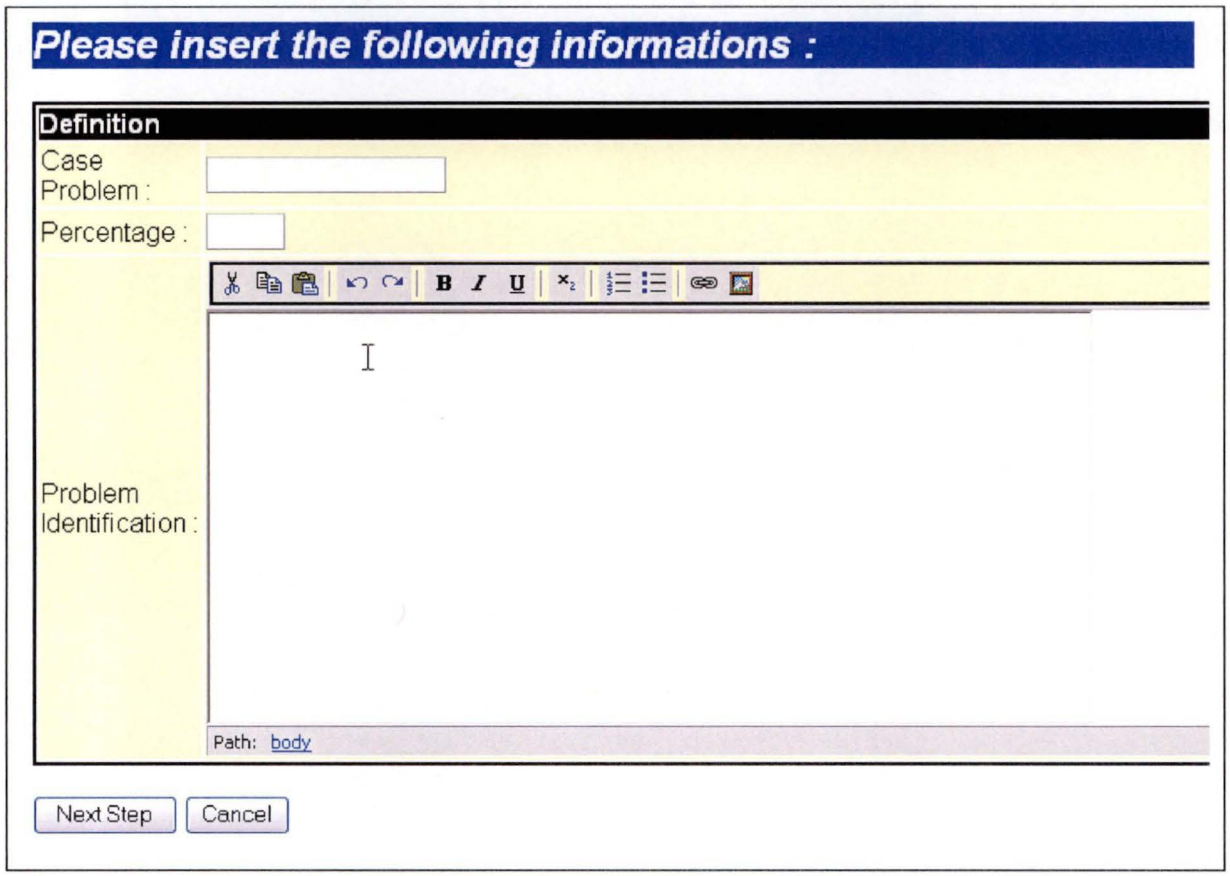


Figure 7.15 *Exemple de formulaire, tiré de CosmicXpert*

7.4.2 Les fichiers XSL

L'utilisation de la technologie XSL, permettant de formater l'affichage de fichiers XML, va assurer le reste du contrôle des contraintes définies par les schémas XML, et même au-delà, puisqu'il est possible d'introduire d'autres contraintes d'ordre conceptuel dans ces fichiers XSL.

La Figure 7.16 illustre ce mécanisme. On peut y observer que l'expert ne pourra pas supprimer l'association existant entre le mot-clé « *Data Group* » et les concepts topologiques « *Data Group* », « *Data Group-MIS* » et « *Data Group-Real Time* ». Cette contrainte est

imposée par le fichier XSL, puisque les cases à cocher pour ces trois concepts topologiques sont désactivées, interdisant toute modification. La contrainte modélisée ici répond au fait qu'on ne peut pas supprimer le lien entre un concept topologique et le mot-clé qui le défini, cette contrainte n'est pas formalisée dans un schéma XML.

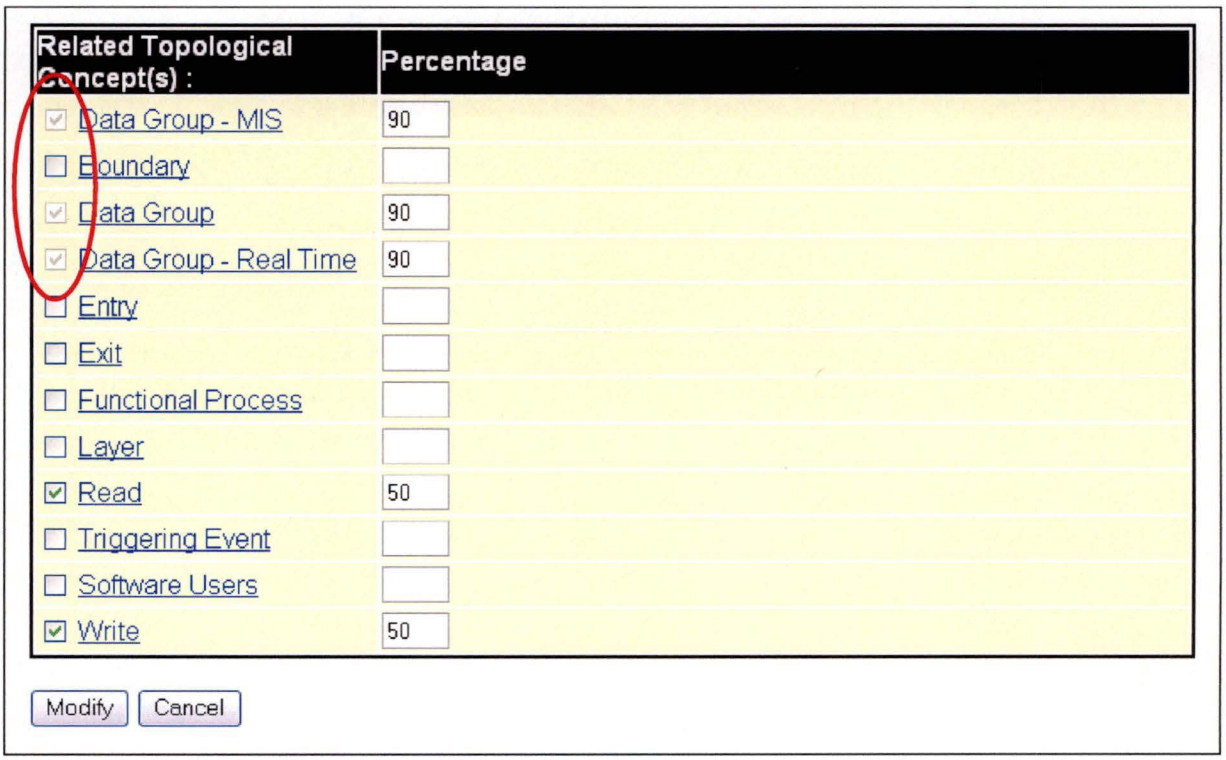


Figure 7.16 *Exemple de contrainte imposée par un fichier XSL*

7.5 Conclusion

Le mécanisme de vérification implémenté pour CosmicXpert répond à ses spécifications. Premièrement, il est aisé de constater que ce mécanisme est effectivement *automatique*, puisque l'expert ne doit jamais intervenir dans le processus de vérification de la base de connaissances. Ceci constitue une avancée importante par rapport au prototype 2 de CosmicXpert développé par Vilz et Gruselin en 2003.

Deuxièmement, ce mécanisme s'appuie sur l'addition des principes de vérification issus d'une part des concepts des bases de données, et d'autres par sur les opportunités offertes par l'interface. Il résulte donc une double vérification complémentaire, assurant au maximum la vérification de la base de connaissances.

Le système développé possède néanmoins quelques limites. En effet, la vérification effectuée par le prototype 3 de CosmicXpert est automatisé et garantit par là le respect des contraintes d'intégrité existantes sur la base de connaissances. Cependant, ces contraintes ne sont pas absolues. De par leur formalisme, les contraintes ne peuvent s'assurer que de la vérification de la présentation et pas de celle du contenu. En d'autres termes, la vérification réalisée est d'ordre syntaxique mais pas sémantique. Il en résulte que les modifications introduites dans la base de connaissances doivent respecter certaines règles, mais qu'il est néanmoins possible d'introduire des concepts farfelus aux yeux des experts.

Par exemple, le système oblige un expert a relier un mot-clé avec au moins un concept topologique mais il ne pose aucune conditions sur le concept topologique en question. De ce fait, un expert pourra relier le mot-clé « *Data Group* » avec le concept topologique « *Boundary* », alors que cette association n'est pas porteuse de sens au regard de la norme COSMIC-FFP.

Chapitre 8 : Illustration

Afin d'illustrer les propos exposés jusqu'à présent, voici le cheminement que doit effectuer un expert désireux d'introduire un nouveau cas problème. Cet exemple nous semble judicieux pour représenter les différentes techniques mises en œuvres dans la partie dédiée aux experts, dans le troisième prototype de CosmicXpert.

Avant de pouvoir entamer cette illustration, il est important de souligner une caractéristique d'utilisation présente à travers toute l'interface de l'expert.
L'idée générale du fonctionnement de l'interface expert est que l'utilisateur doit sélectionner dans un premier temps le concept sur lequel il désire travailler, avant de préciser la fonction qu'il veut effectuer sur celui-ci.
Dans le cas de l'ajout d'un nouveau concept, c'est le concept parent du nouveau concept qui doit être sélectionné. Le parent sélectionné sera pris comme un paramètre « implicite ».
Seul l'ajout d'un mot-clé et d'un concept topologique ne fonctionne pas selon ce modèle, ceux-ci n'ayant pas de concept parent. Deux menus d'accès direct à ces fonctions ont dès lors été ajoutés.

L'ajout d'un cas problème passe donc par plusieurs étapes qui sont :

- la sélection du concept topologique auquel le cas problème sera rattaché,
- la sélection de la fonction à effectuer,
- l'introduction des informations concernant le cas problème proprement dit,
- l'introduction des informations concernant la recommandation attachée au nouveau cas problème,
- fin de l'opération.

8.1 Sélection du concept topologique

Les cas problèmes étant directement reliés aux concepts topologiques, il faut tout d'abord sélectionner celui auquel on désire rattacher le nouveau cas problème. Cette opération peut être effectuée à partir de l'un des deux arbres comprenant la liste des concepts topologiques.

La Figure 8.1 représente l'arbre « Concepts topologiques – Mots-clés – Cas problèmes » (l'autre arbre possible étant l'arbre « Concepts topologiques – Thèmes »).

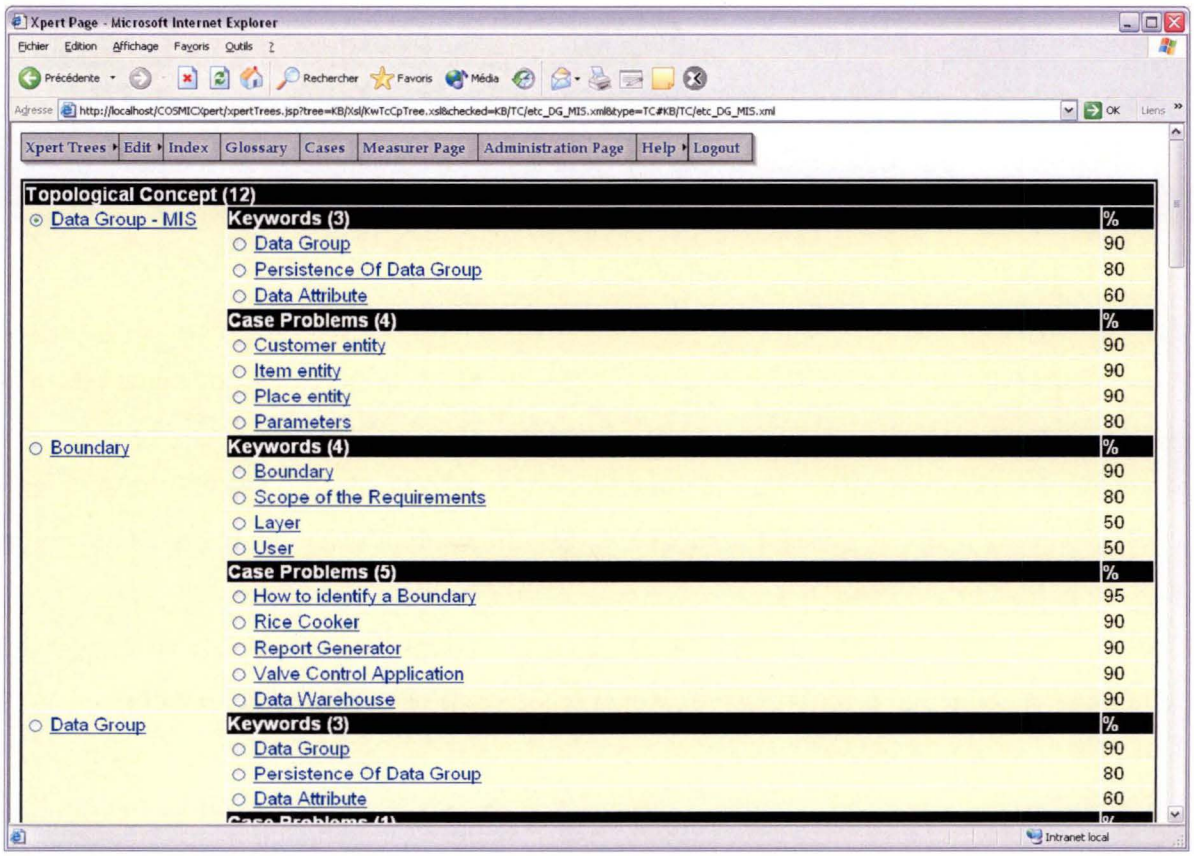


Figure 8.1 *Sélection d'un concept topologique*

8.2 Sélection de la fonction

La fonction à effectuer est sélectionnée grâce au menu « `Edit` » du menu dynamique[11] (cf. Figure 8.2).

[11] Les pages affichées pouvant être très longues dans certains cas, ce menu suivra le défilement de la fenêtre.

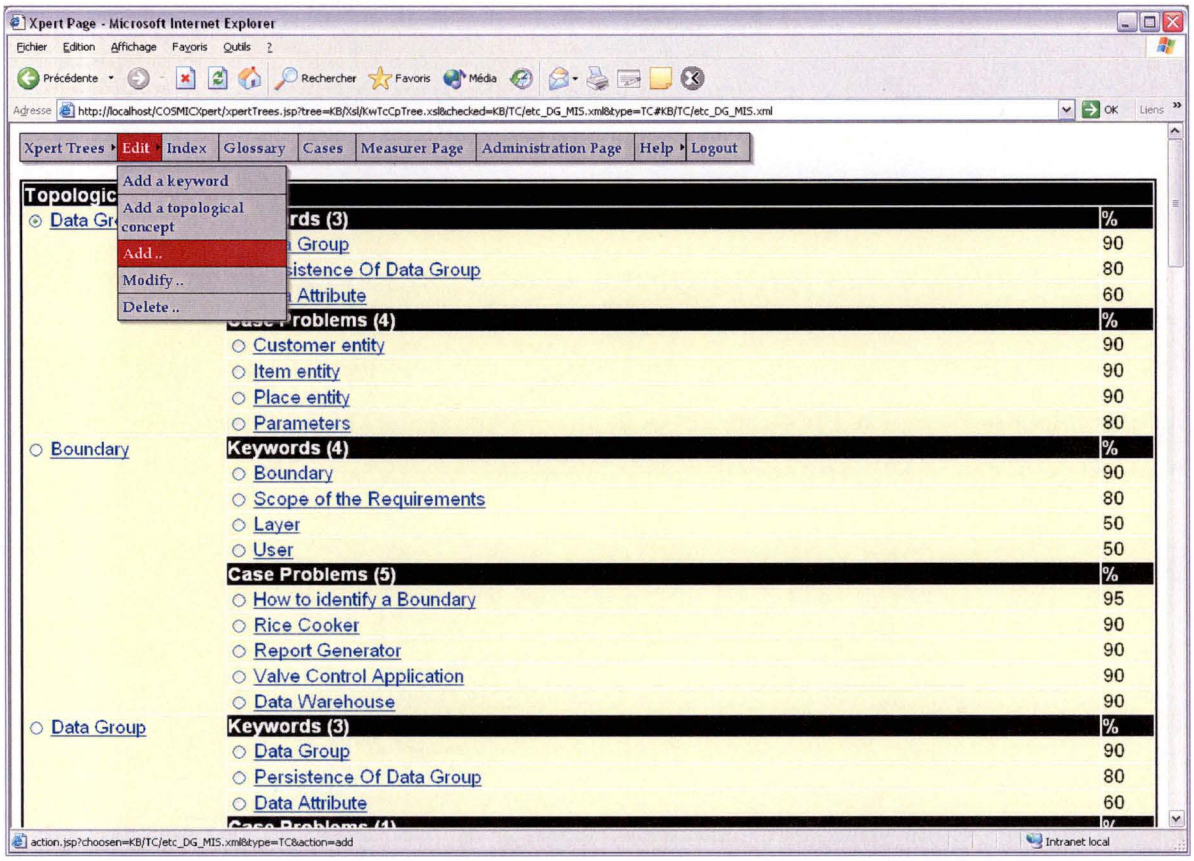


Figure 8.2 *Sélection de la fonction à effectuer (1 de 2)*

Ce menu réagit au type de la sélection. Dans notre cas, il s'agit d'un concept topologique. Le système proposera donc à l'utilisateur de choisir entre l'ajout d'un cas problème ou l'ajout d'un thème. En effet, comme le montre la hiérarchie des concepts à la Figure 4.8, les concepts topologiques chapeautent à la fois les thèmes et les cas problèmes.

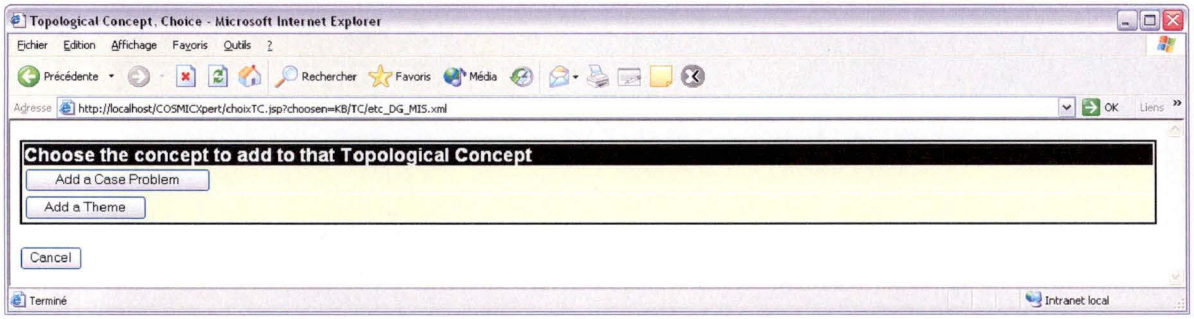


Figure 8.3 *Sélection de la fonction à effectuer (2 de2)*

Dès l'instant où la fonction a effectuer a été sélectionnée, le système demande la pose d'un verrou sur cette connaissance. Si le concept topologique n'est pas encore utilisé par un autre

expert, c'est-à-dire s'il n'existe pas encore de verrou sur cette connaissance, le système la verrouille. Sinon, le système prévient l'utilisateur qu'il ne peut poursuivre, la connaissance étant en cours d'utilisation, et lui demande de patienter quelques instants.

8.3 Introduction des informations concernant le cas problème

Les nouvelles informations introduites par l'expert doivent vérifier les contraintes imposées sur les connaissances. En l'occurrence, pour un cas problème, elles doivent répondre à la chartre de structuration illustrée par la figure suivante :

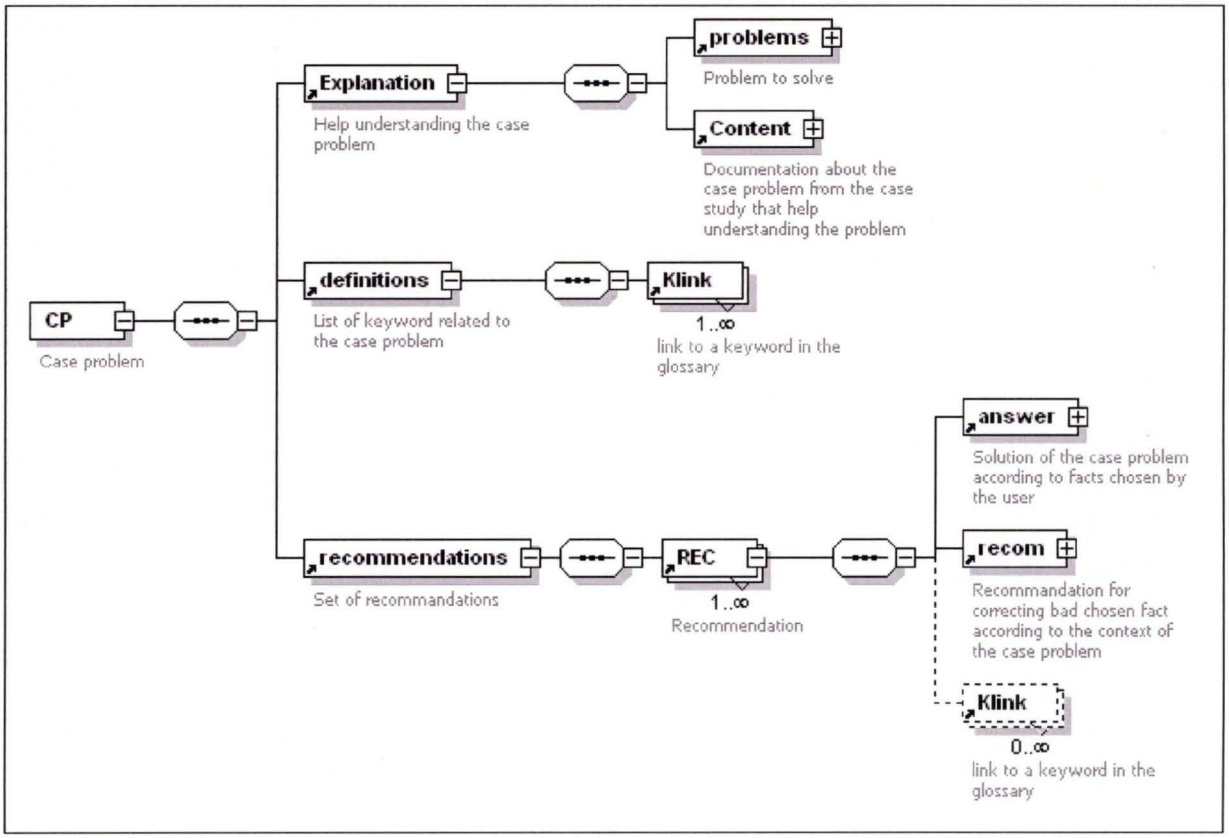


Figure 8.4 *Schéma d'un document cas problème*

Nous pouvons dès lors, voir quelles informations sont nécessaires pour l'ajout d'un cas problème. Ces différentes informations, ainsi que leurs attributs, sont explicitées dans le Tableau 8.1.

Tableau 8.1 *Description d'un document "Cas Problème"*

Élément	Description	Type de données	Attributs	
CP	Délimite le document Cas Problème		tc	Référence au concept topologique parent
			name	Nom du cas problème
			cs	Référence à l'étude de cas
Explanation	Délimite l'explication du cas problème			
problems	Explication sommaire du cas problème	TextHTML		
Content	Explication détaillée du cas problème	TextHTML		
definitions	Délimite une série de mots-clés relatifs au cas problème			
Klink	Représente un lien vers un mot-clé du glossaire		idref	Identifiant du mot-clé lié dans le glossaire
recommendations	Délimite la recommandation associée au cas problème			
REC	Délimite un cas de recommandation		id	Identifiant du cas de recommandation
			mincf	Pourcentage
			maxcf	Pourcentage
answer	Délimite la formulation du cas de recommandation	TextHTML		
recom	Délimite l'explication du cas de recommandation	TextHTML		
Klink	Représente d'éventuels liens vers un mot-clé du glossaire		idref	Identifiant du mot-clé lié dans le glossaire

Par souci de clarté pour l'expert, l'acquisition de ces informations se déroulera sur plusieurs pages qu'il sera invité à compléter au fur et à mesure. Les sections suivantes décrivent chacune de ces pages ainsi que les exceptions qui pourraient s'en suivre.

8.3.1 Introduction de la première partie des informations

Dans la première page, l'expert doit introduire les informations relatives aux champs « `CP` » et « `Explanation` » susmentionnés, ainsi que le pourcentage du lien entre le cas problème et son concept topologique. Cette dernière information ne se retrouvera pas telle quelle dans le fichier du nouveau cas problème, mais bien dans le fichier « `xpert.xml` » qui contient en particulier les associations entre un concept topologique et les cas problèmes.

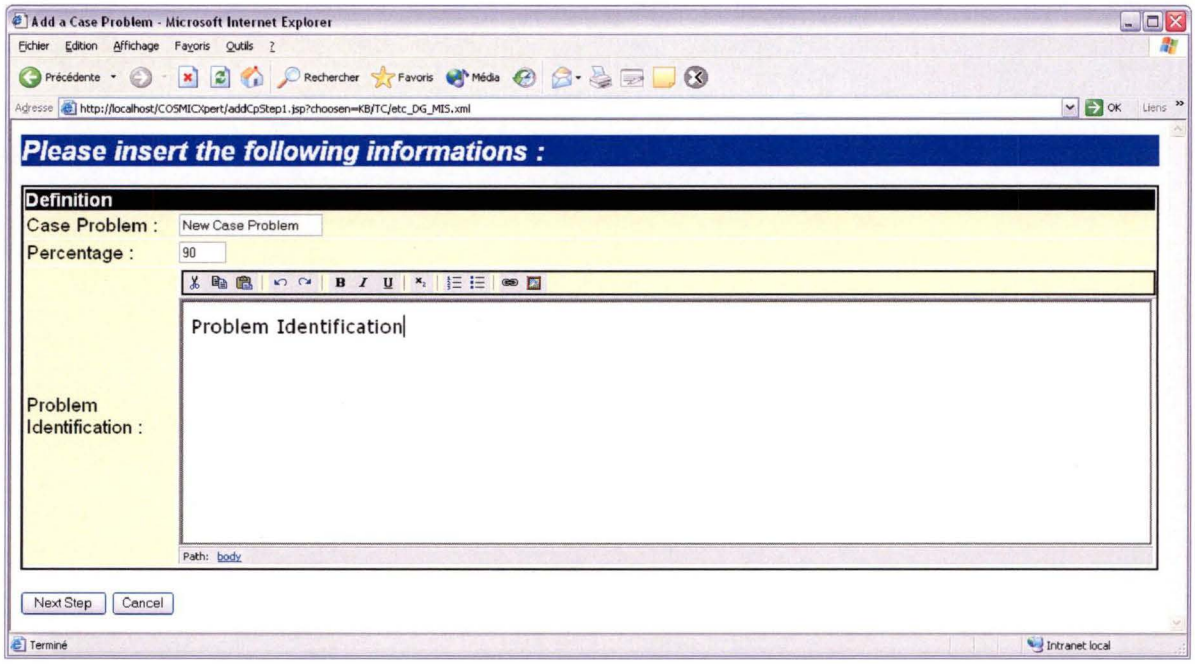


Figure 8.5 *Introduction de la première partie des informations concernant un nouveau cas problème*

8.3.1.1 Alternative : Un ou plusieurs champs n'ont pas été remplis

Tel que précisé à la section 7.4.1, plusieurs vérifications sont effectuées via l'interface elle-même grâce aux formulaires proposés à l'expert. Ces vérifications sont basées sur la nature des informations introduites dans les champs du formulaire. Ainsi, comme l'illustre la Figure 8.6, si l'expert laisse des champs vides, alors que ceux-ci sont obligatoires, ou s'il introduit une valeur incorrecte par rapport à ce qu'il lui est demandé, il en sera directement informé par un message d'erreur lui indiquant ce que le système attend comme type de données.

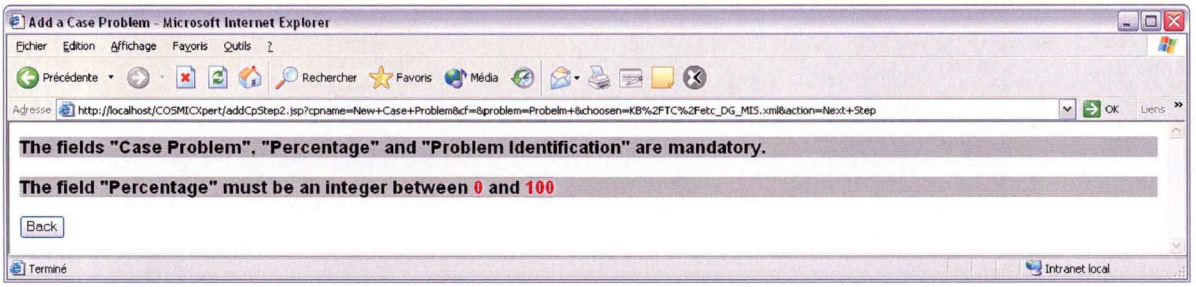


Figure 8.6 *Alternative : un ou plusieurs champs n'ont pas été remplis*

Nous verrons plus tard dans ce chapitre que ce type de vérification permet aussi d'éliminer d'autres types d'erreurs.

8.3.2 Introduction de la seconde partie des informations

La section 7.4.2 mentionne également que l'interface expert permet une vérification basée sur l'utilisation de la technologie XSL, permettant de formater l'affichage de fichiers XML en HTML, ne proposant ainsi que les entrées valides à l'expert. Ce mécanisme est illustré à la Figure 8.7, lors de l'introduction de la seconde partie des informations relatives au nouveau cas problème.

Dans un premier temps, l'expert remplit une explication plus détaillée du nouveau cas problème. Cette partie est similaire à l'introduction des premières informations.

C'est dans la deuxième partie de la page qu'intervient la technologie XSL. En effet, l'expert est invité à sélectionner l'étude de cas auquel sera rattaché le cas problème. Evidemment, seuls les études de cas existantes sont proposées. De plus, un cas problème n'étant rattaché qu'à une et une seule étude de cas, nous avons recours aux « *Radio Button* », définit par la norme HTML et ne permettant la sélection que d'un seul concept à la fois.

Pour le reste de la page, l'expert sélectionne les mots-clés ainsi que les noeuds relatifs, s'il le désire. De nouveau, la technologie XSL va interdire à l'expert d'effectuer certaines opérations. Un cas problème étant rattaché à un seul concept topologique, et ce dernier possédant une définition enregistrée via un mot-clé, il sera impossible à l'expert de déselectionner ce mot-clé (cf. le rond rouge de la Figure 8.7). Par contre, il pourra sélectionner autant de mots-clés et de noeuds qu'il le désire, via l'utilisation de « *Check Boxes.* »

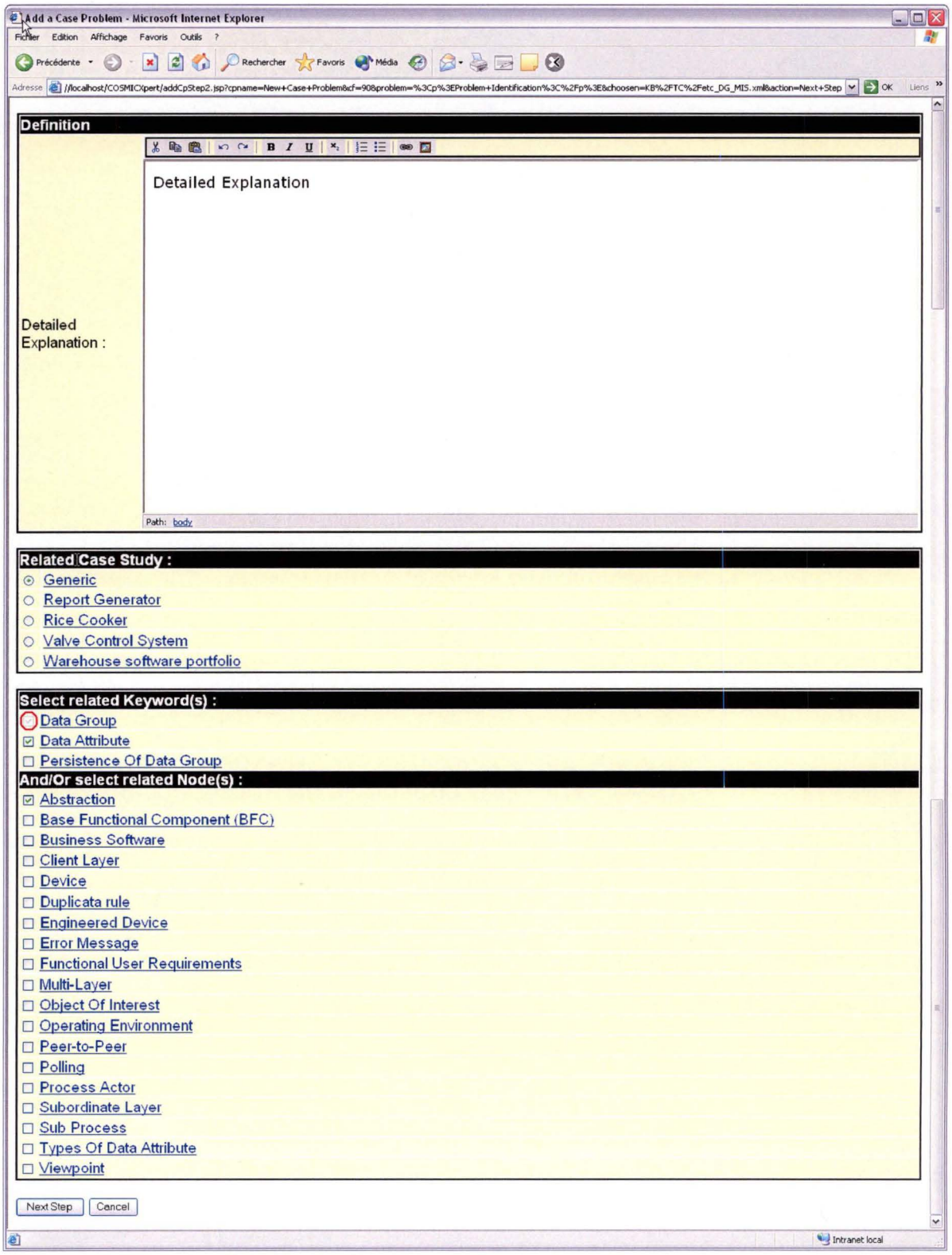


Figure 8.7 *Introduction de la seconde partie des informations concernant un nouveau cas problème*

8.4 Introduction des informations concernant la recommandation attachée au nouveau cas problème

Le rôle de l'interface que nous avons développée ne s'arrête pas uniquement à la vérification des informations. Elle fournit également une aide à l'expert pour l'introduction d'informations dans le sens où elle propose des informations préexistantes. Ceci est illustré dans la suite du processus de création d'un cas problème. En effet, lors de ce processus, l'expert doit également entrer des informations relatives à une nouvelle recommandation[12] qui sera associée au nouveau cas problème. Pour ce faire, le système propose à l'expert de sélectionner une recommandation préexistante qu'il pourra modifier à sa guise. Naturellement, par souci de cohérence, il ne sera proposé à l'expert que des recommandations ayant un certain lien logique avec le nouveau cas problème. Il s'agira donc des recommandations associées aux cas problèmes ayant le même concept topologique comme concept parent. Ce mécanisme est illustré par les figures suivantes (Figure 8.8 à Figure 8.20).

[12] Il est important de noter que nous désignons par « recommandation » l'ensemble des cas de recommandations attachées à un cas problème.

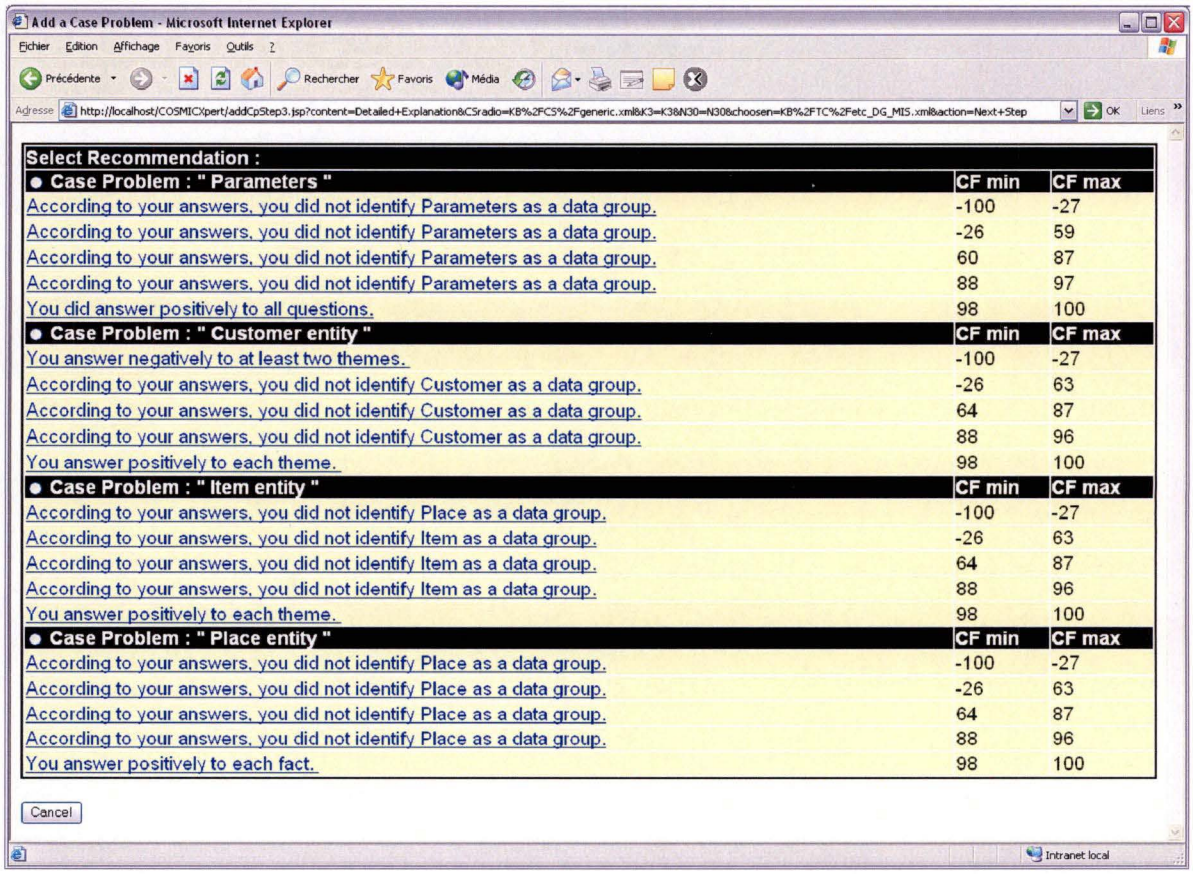


Figure 8.8 *Sélection d'une recommandation existante*

La Figure 8.8 propose à l'expert les quatre recommandations existantes pour le concept topologique « `Data Group - MIS` », qui est le concept parent du nouveau cas problème. Il suffit à l'expert d'en sélectionner une, en vue de la modifier comme le montre la Figure 8.9. Cette dernière page lui permet d'effectuer ces opérations de modifications. S'il ne désire pas transformer la recommandation existante, il peut directement terminer le processus d'insertion d'un nouveau cas problème, ajoutant ainsi celui-ci à la base de connaissances.

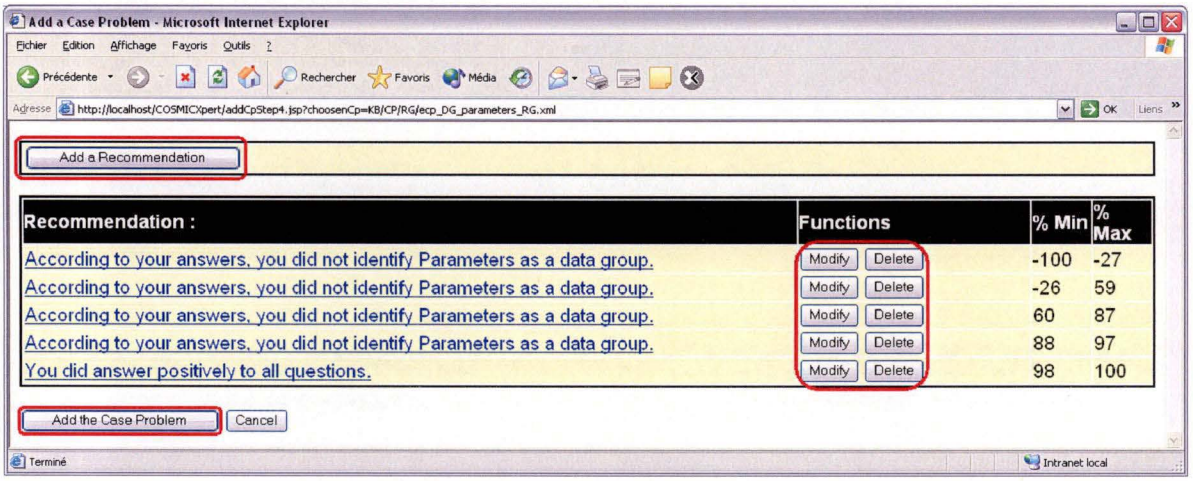


Figure 8.9 *Modification de la recommandation existante*

Toute recommandation sélectionnée peut être modifiée. De plus, tous les cas de recommandation d'une même recommandation peuvent faire l'objet de changements. Il suffit pour cela à l'expert de sélectionner l'opération à réaliser sur les cas de recommandation qu'il souhaite modifier.

8.4.1 Ajout d'un cas de recommandation

Une fois l'opération désirée sélectionnée, l'expert arrive sur une nouvelle page. Dans cette section nous illustrons l'opération d'ajout d'un nouveau cas de recommandation. La section suivante, quant à elle, sera consacrée à l'opération de suppression d'un cas de recommandation[13].

8.4.1.1 Introduction de la première partie des informations

Nous retrouvons à la Figure 8.10 une fenêtre classique pour l'ajout d'informations relatives à un concept particulier. Comme l'illustrent les figures suivantes (Figure 8.11 à Figure 8.14), ce sont surtout les erreurs qui varient en fonction du concept en question.

[13] L'opération de modification d'un cas de recommandation étant fort semblable à celle d'ajout, nous n'illustrerons pas cette fonctionnalité ici, afin de rester concis.

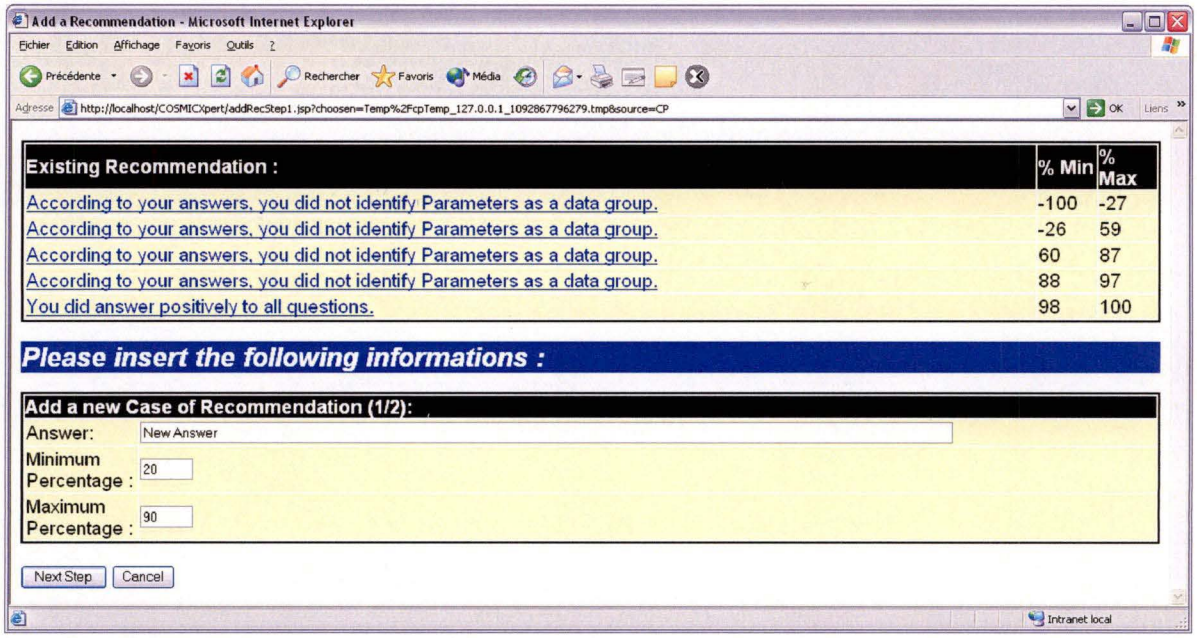


Figure 8.10 *Introduction de la première partie des informations concernant un nouveau cas de recommandation*

8.4.1.2 Alternative : Les valeurs introduites sont incorrectes

La Figure 8.11 montre le type d'erreur que l'expert pourrait commettre en introduisant ses informations. De même, les figures suivantes montrent les messages d'erreurs qui pourraient en résulter.

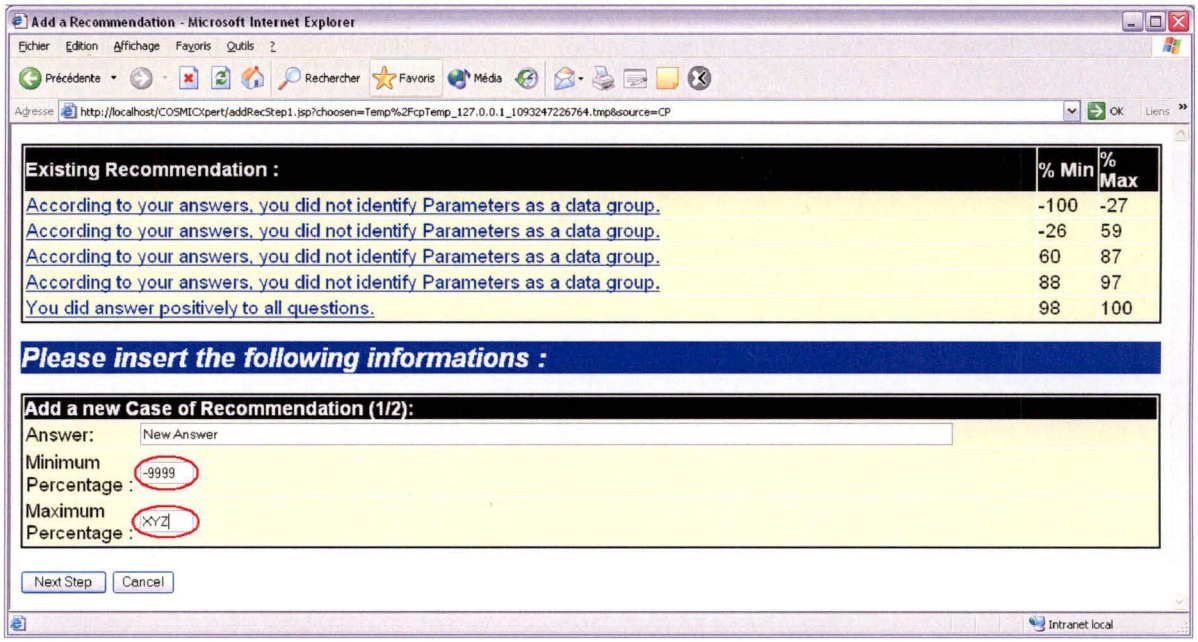


Figure 8.11 *Alternative : les valeurs introduites sont incorrectes*

Les deux premières erreurs détectées (Figure 8.12 et Figure 8.13) font référence à une cohérence, par rapport au type de données qui sont demandés. Ces messages d'erreurs peuvent survenir si l'expert introduit du texte alors que l'information demandée est de type « entier », ou si l'expert introduit des entiers ne rentrant pas dans la plage des valeurs autorisées.

La troisième erreur illustrée par la Figure 8.14, quant à elle, est d'ordre logique. En effet, de façon à ce que CosmicXpert puisse proposer des cas de recommandations différents aux mesureurs en fonction de leurs réponses aux thèmes, il n'est pas permis à un expert d'introduire un cas de recommandation allant de -100 à 100, puisque celui-ci serait alors présenté aux experts dans tous les cas.

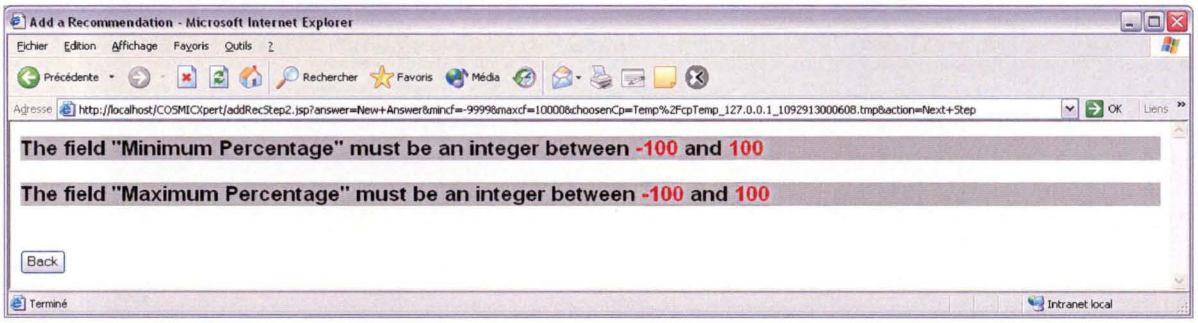


Figure 8.12 *Alternative : Les pourcentages introduits ne sont pas des entiers (compris entre -100 et 100)*

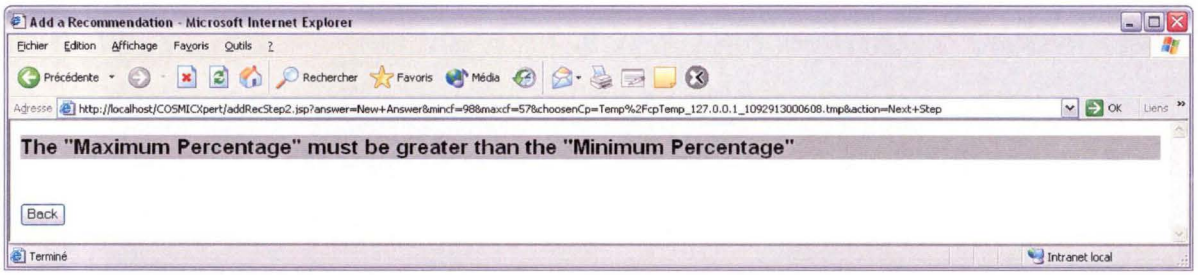


Figure 8.13 *Alternative : Le pourcentage minimum introduit est supérieur au pourcentage maximum introduit*

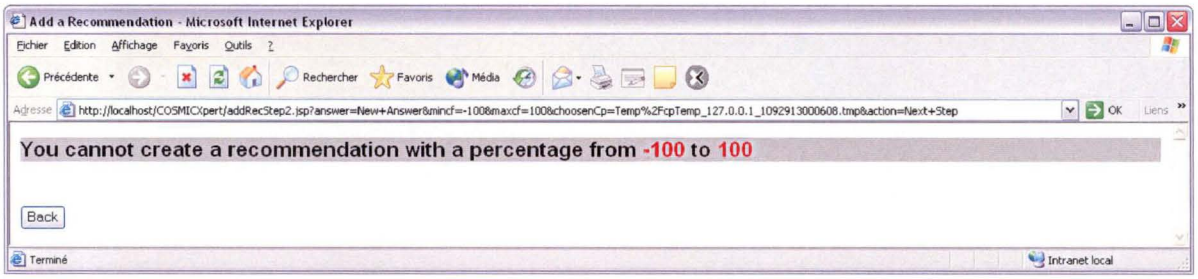


Figure 8.14 *Alternative : Impossibilité de créer un cas de reommandation de -100 jusque 100*

8.4.1.3 Introduction de la deuxième partie des informations

L'introduction de la deuxième partie des informations concernant un nouveau cas de recommandation n'apporte pas de renseignements nouveaux au point de vue des mécanismes de vérification. Le système demande simplement à l'expert d'introduire une description pour le cas de recommandation et de l'associer avec des mots-clés et/ou nœuds.

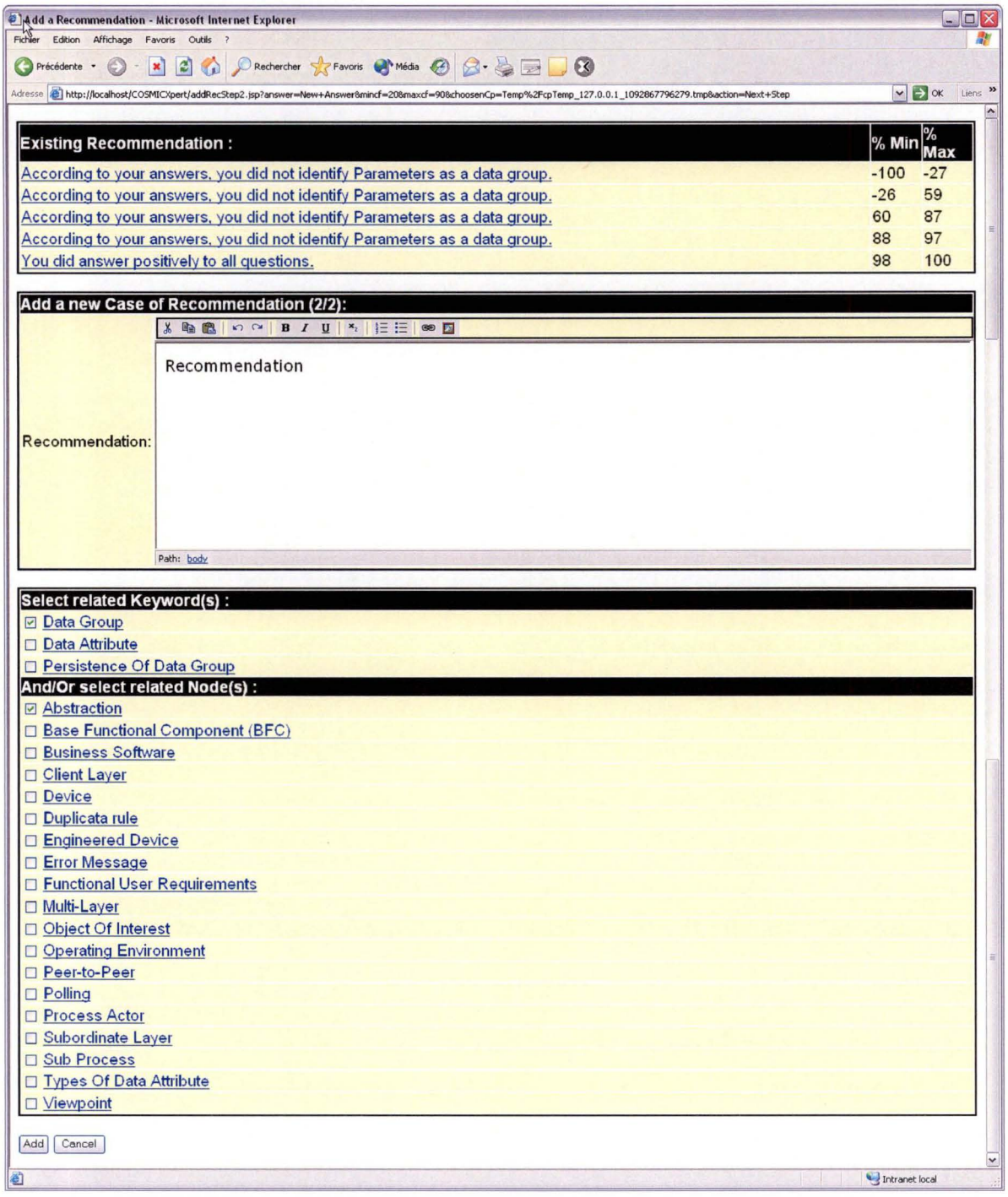


Figure 8.15 *Introduction de la seconde partie des informations concernant nouveau cas de recommandation*

8.4.2 Confirmation

L'ajout d'un cas de recommandation (ainsi que la modification d'un de ceux-ci) n'est pas sans influence sur les autres cas de recommandation de la recommandation existante. En effet, les pourcentages doivent s'étaler sur une échelle de -100 à 100 et être contigus. De ce fait, les

pourcentages des autres cas de recommandation doivent être modifiés en conséquence. Si le nouveau cas de recommandation recouvre complètement la plage d'un autre cas de recommandation, ce dernier sera totalement supprimé. Toutes ces modifications se font de manière automatique et sont proposées à l'expert afin qu'il puisse confirmer les modifications qu'il a apporté (cf. Figure 8.16).

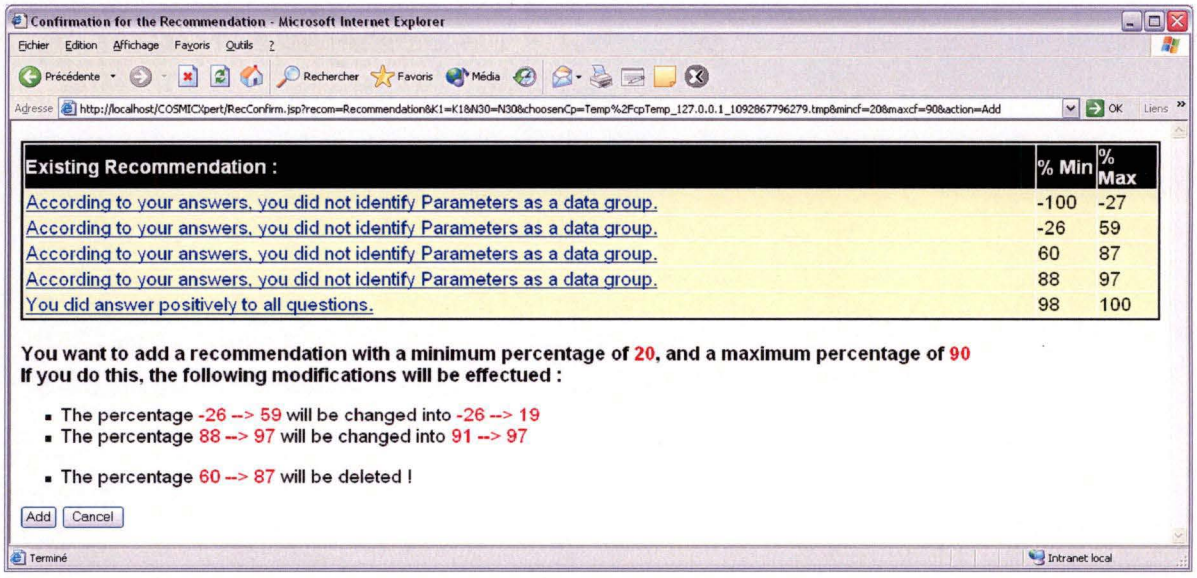


Figure 8.16 *Confirmation pour l'ajout d'un nouveau cas de recommandation*

8.4.3 Autre(s) modification(s) de la recommandation existante

Une fois toutes ces informations introduites pour l'ajout d'un nouveau cas de recommandation, une fenêtre affiche l'état de la nouvelle recommandation. A partir de celle-ci, l'expert pourra effectuer autant de nouvelles modifications qu'il le désire.

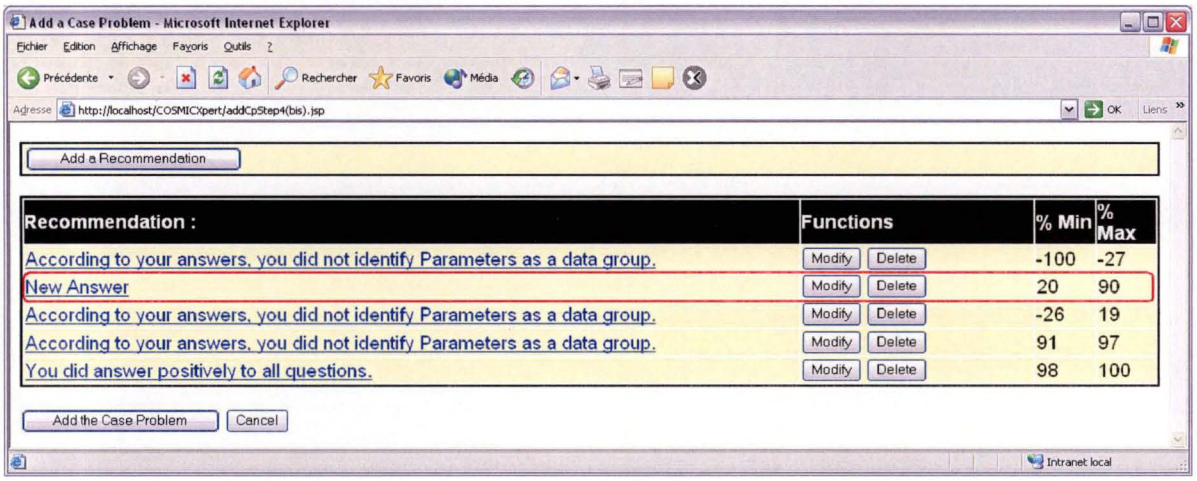


Figure 8.17 *Nouvelle modification de la recommandation existante*

8.4.4 Suppression d'un cas de recommandation

La suppression d'un cas de recommandation implique également des modifications au niveau des pourcentages des autres cas de recommandation. Mais, contrairement au processus d'ajout ou de modification d'un cas de recommandation, ces changements ne peuvent se faire sans l'aide de l'expert. Une nouvelle fenêtre est alors proposée à l'expert afin qu'il puisse y introduire les nouveaux pourcentages.

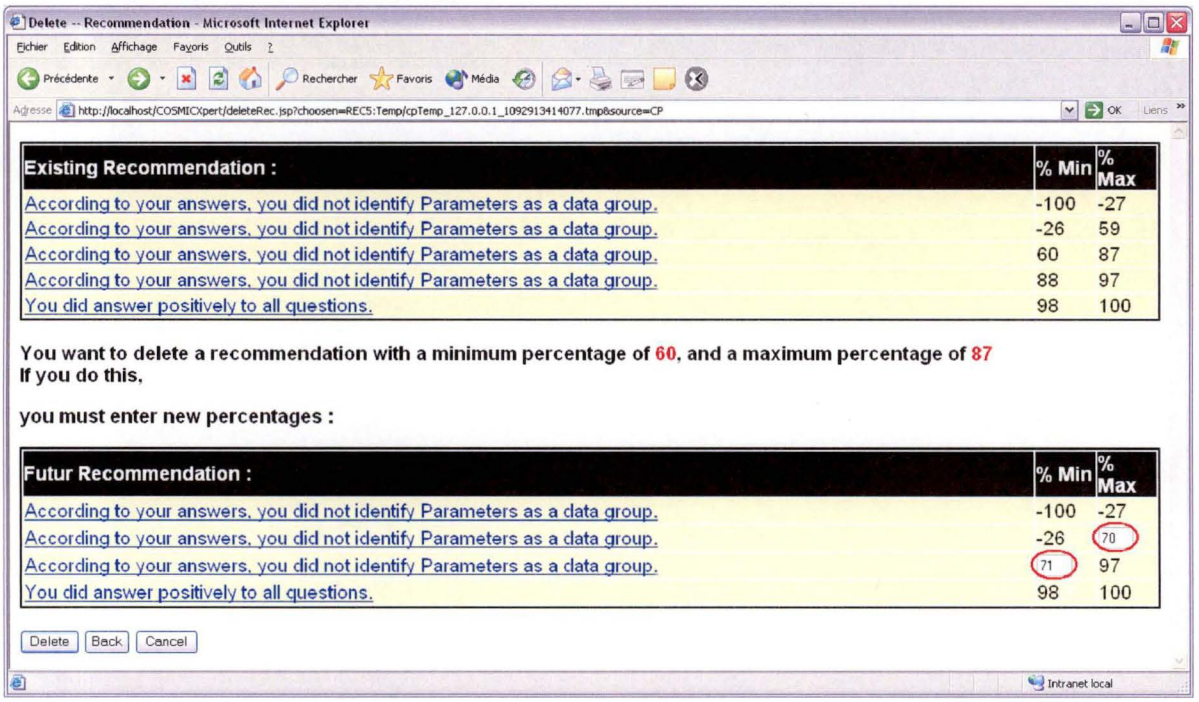


Figure 8.18 *Suppression d'un cas de recommandation*

Si les nouveaux pourcentages introduits ne sont pas valides (c'est-à-dire ne sont pas contigus), un message d'erreur sera affiché à l'expert (Figure 8.19).

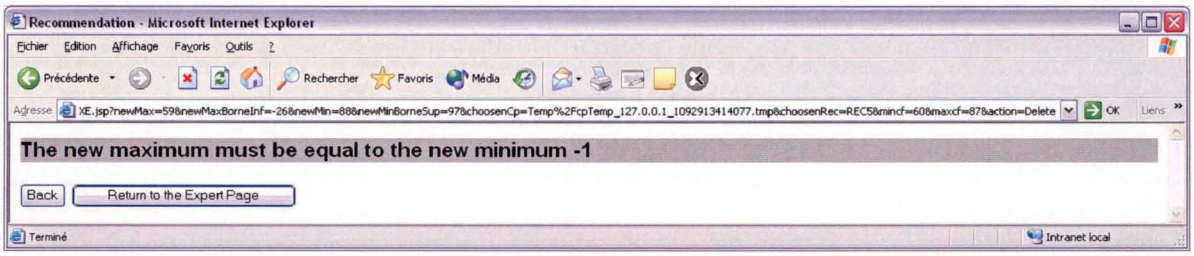


Figure 8.19 *Alternative : les pourcentages introduits ne sont pas contigus*

Si l'expert demande la suppression d'un cas de recommandation, alors que la recommandation n'en compte plus que deux, la suppression sera refusée, de façon à éviter qu'un cas de recommandation soit présenté en toute circonstance à un mesureur, peu importe ses réponses aux thèmes. Cette contrainte a été imposée afin d'assurer un minimum de signification pour le mesureur. Le message d'erreur qui en résulte est présenté à la Figure 8.20.

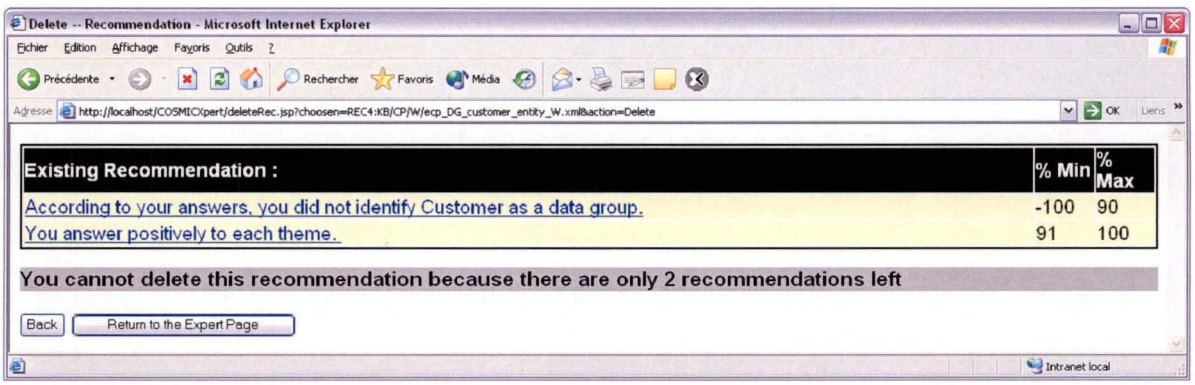


Figure 8.20 *Alternative : il n'y a que deux cas de recommandation*

Une fois cette opération effectuée, l'expert est de nouveau renvoyé vers la page récapitulative sur l'état de la recommandation actuelle (Figure 8.17). Il peut ensuite finalement conclure le processus d'ajout d'un nouveau cas problème.

8.4.5 Fin de l'opération

Jusqu'à présent, l'interface a permis d'accepter uniquement des informations valides. C'est une certaine facette du processus de vérification. C'est lors de cette dernière étape que ces

informations vont effectivement être enregistrées dans la base de connaissances. Il est à noter que l'expert n'intervient plus lors de cette phase. Cette dernière est automatique et les différentes opérations se déroulent de manière invisible aux yeux de l'expert. Seuls un message de succès (Figure 8.21) ou d'erreur est affiché.

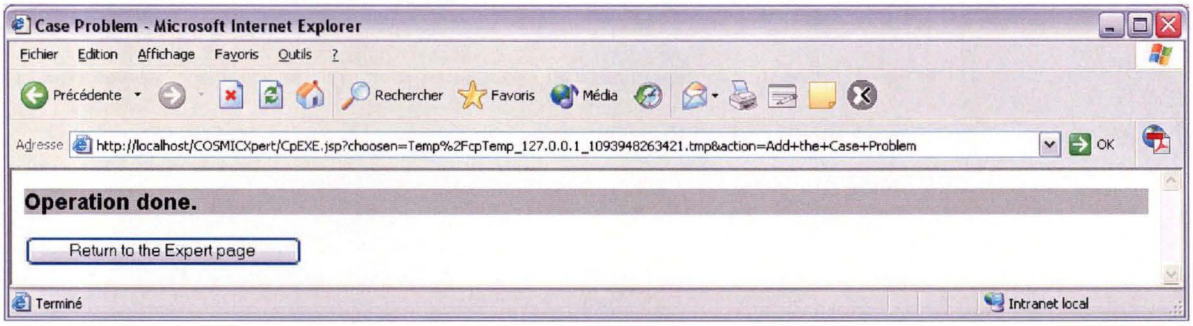


Figure 8.21 *Opération effectuée*

Néanmoins, afin de garder une base de connaissances cohérente, il faut encore que ces informations soient correctement enregistrées. C'est pourquoi le système fait appel à un mécanisme de transaction afin de réguler la concurrence d'accès aux fichiers, ainsi que de réaliser la protection contre les incidents.

Dans le cadre de notre exemple, les fichiers impliqués sont le fichier du nouveau cas problème et le fichier « `xpert.xml` » qui enregistre la magnitude du lien existant entre ce nouveau cas problème et son concept topologique parent.

Il faut donc interdire l'accès en écriture sur ces deux fichiers pendant toute la durée de la transaction et libérer ces verrous à la fin de celle-ci.

Pour rappel, nous utilisons la 3ème solution proposée dans le chapitre 7. Cette solution prône un double de mécanisme de verrouillage :

- Un verrou est posé dès le début de la requête d'ajout sur la *connaissance* visée. Dans notre cas il s'agit de la connaissance représentant le concept topologique parent. Il est donc impossible à tout autre expert d'effectuer une modification (en ce compris une suppression) sur ce concept topologique pendant toute la durée de l'ajout du nouveau cas problème.
- Un verrou est posé sur chacun des *fichiers* auxquels le système devra accéder pendant la transaction, empêchant ainsi toutes interférences avec les requêtes d'autres experts. Dans notre cas il s'agit donc des deux fichiers

susmentionnés, à savoir le fichier du nouveau cas problème et le fichier « `xpert.xml` ».

Entre ces étapes de pose et de relâche des verrous, la transaction effectue toutes les opérations nécessaires à l'enregistrement des données dans la base de connaissances.

Dans un premier temps, des fichiers temporaires sont créés. Ces fichiers sont la copie conforme des fichiers sources sur lesquels la transaction doit travailler[14]. Les fichiers temporaires portent ici les noms suivants :

- `KB;xpert.xml@127.0.0.1@1093604476843.log`
- `KB;CP;G;ecp_DG_how_to_identify_a_data_group_G.xml@127.0.0.1@1093604476843.log`

Les noms des fichiers temporaires sont constitués du nom de leur fichier source, augmenté du chemin d'accès de ces fichiers dans la base de connaissances. Ils contiennent également un identifiant de transaction composé de l'adresse IP de la machine de l'expert et d'un nombre unique généré par le serveur. Cet identifiant est nécessaire dans un contexte distribué.

Les changements sont appliqués sur ces fichiers temporaires et non directement sur les fichiers sources. Ceci permet une récupération plus aisée en cas d'incident. En effet, si un incident se produit pendant que les fichiers sont en cours de modification, la base de connaissances ne devra pas être restaurée, puisque seuls les fichiers temporaires auront été modifiés. Si ces opérations de modifications se sont déroulées correctement, le contenu des fichiers temporaires est recopié dans les fichiers sources.

Les fichiers temporaires sont ensuite déplacés dans un répertoire de sauvegarde (dossier « *Backup* »), afin de constituer une copie de récupération. S'il existait une version de sauvegarde antérieure correspondant aux fichiers temporaires, ceux-ci seront remplacés par les nouveaux fichiers. Seules les dernières versions de chaque fichier sont donc conservées, optimisant la récupération après un incident. Ce mécanisme correspond à un « *journal des images après* ». Comme nous l'avons définit en 3.3.3, ce journal contient la copie des pages après leur modification. Permettant de récupérer la base de connaissances après un incident majeur, la détruisant totalement ou en partie.

---

[14] Lors de l'ajout d'un cas problème, le fichier source utilisé est le fichier du cas problème dont la recommandation a servi de base dans le processus de création d'une nouvelle recommandation (illustré à la section 8.4).

Conclusion

Le domaine de l'Intelligence Artificielle a connu depuis sa création des phases prospères et des phases plus difficiles. A l'origine, tous les espoirs étaient permis. Durant les années 1990, on s'était largement questionné sur la faisabilité de systèmes à base de connaissances utiles. A l'heure actuelle, les entreprises semblent toujours intéressées par l'introduction de SBC dans leur structure. Cependant, pour que cet intérêt reste soutenu, il faudrait à notre avis, corriger certaines faiblesses des SBC. Une des constatations de la faiblesse de tels systèmes concerne le vocabulaire ou encore l'ontologie du domaine.

La définition claire d'une ontologie apparaît donc comme indispensable. Cette condition n'est cependant pas suffisante. Il subsiste encore un problème de vérification car l'existence d'une ontologie ne garantit pas que le contenu de la base de connaissances soit cohérent. On peut seulement affirmer que l'ontologie permet la modélisation du domaine. Cette opération est réalisée par l'ingénieur de la connaissance, qui est investit d'une lourde tâche, puisqu'il doit modéliser les connaissances des experts du domaine et les introduire dans le système. Il serait préférable que les experts eux-mêmes soient en mesure d'introduire leurs connaissances, même si l'ingénieur de la connaissance reste toujours responsable de la modélisation du domaine. Cette tâche peut être facilitée par l'ajout d'une interface et de procédures. L'ingénieur de la connaissance peut être supprimé du processus de développement et de maintien de la base de connaissances, à condition que ces procédures s'accompagnent de mécanismes de la vérification de la cohérence des informations.

Le projet de développement de CosmicXpert reflète bien cette problématique. En effet, il a été possible de créer un SBC pour COSMIC-FFP, puisque l'ontologie du domaine a été clairement définie. La création d'un premier prototype a donc été réalisable. Celui-ci ne contenait aucun mécanisme de vérification de la base de connaissances, entraînant des difficultés croissantes de maintenance du système. Le second prototype se veut être une réponse à cette faiblesse. Il répond en partie au problème, en formalisant les connaissances de la base et en imposant sur celles-ci un ensemble de contraintes. Malgré cela, il reste incomplet puisqu'il n'autorise pas les modifications des connaissances. Le troisième prototype tente de

réaliser une vérification automatique. Il autorise les experts du domaine à apporter eux-mêmes des changements au sein de la base de connaissances, leur permettant ainsi d'assurer le rôle de l'ingénieur de la connaissance, tout en assurant le maintien de la vérification de la base.
La vérification est dite automatique, car elle est effectuée à l'insu des utilisateurs lors l'introduction de changements dans la base de connaissances de CosmicXpert. Cette vérification est réalisée à travers deux mécanismes particuliers.
Dans un premier temps, l'interface se charge de garantir le respect des contraintes imposées par le formalisme de la base de connaissances.
Dans un second temps, l'utilisation de concepts issus des bases de données permet de garantir le respect des propriétés A.C.I.D, à travers différents modules de type gestion de la concurrence, gestion des incidents, entre autres.
Ces deux niveaux complémentaires permettent ensemble de réaliser une vérification qui se veut des plus complète.

Dans les travaux futurs, il serait intéressant d'étudier si ce mécanisme de vérification automatique, développé pour CosmicXpert, est adaptable à d'autres domaines. Nous savons cependant que l'ontologie du nouveau domaine doit être parfaitement identifiée avant de pouvoir réaliser cette adaptation, sans quoi la formalisation des connaissances serait impossible.

Notre contribution à la vérification de CosmicXpert peut également être complétée. En effet, la vérification réalisée est limitée par le formalisme utilisé, ne permettant qu'une vérification syntaxique des connaissances. Il pourrait être intéressant, lors de travaux futurs, de développer un mécanisme permettant de lier l'ontologie de COSMIC-FFP au système lui-même, afin de réaliser une vérification sémantique. Cette dernière vérification pourrait permettre la garantie d'un système où les connaissances sont cohérentes mais aussi où elles ont une certaine signification au regard de la théorie de la mesure fonctionnelle.

Annexes

Annexe 1 : Charte de structuration

1.1 Structuration des concepts et de leur contenu

La structure des différents fichiers XML est basée sur l'utilisation d'un type défini dans un document XSD annexe (global.xsd). Ce type de données a pour but de retrouver dans la vue créée par le XSL, les paragraphes, listes et images, en rendant possible la représentation de ces entités en XML. Pour cela, il a été défini d'utiliser un type de données appelé *TextHTML*. Nous utiliserons ce type de données dans la description de tous les types de documents. Ce type est représenté à la Figure 1.22.

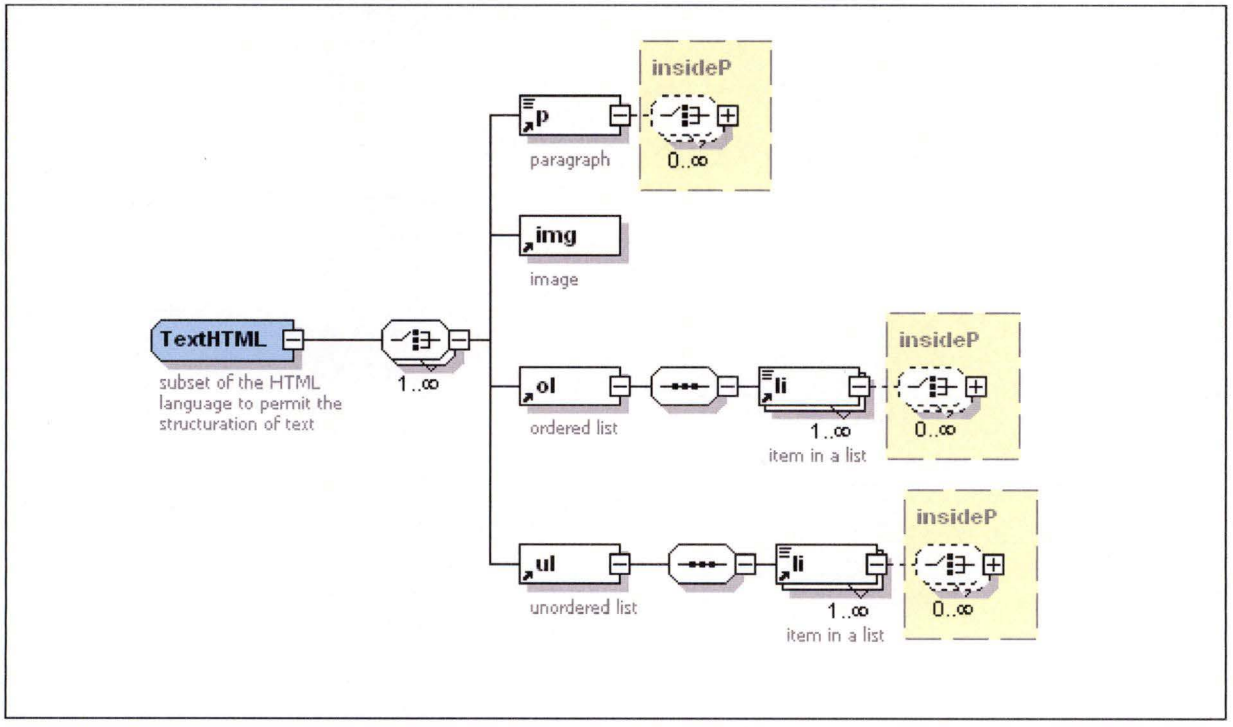


Figure 1.22 *Type « TextHTML »*

Ce type inclus un autre type, à savoir « *InsideP* », illustré à la Figure 1.23.

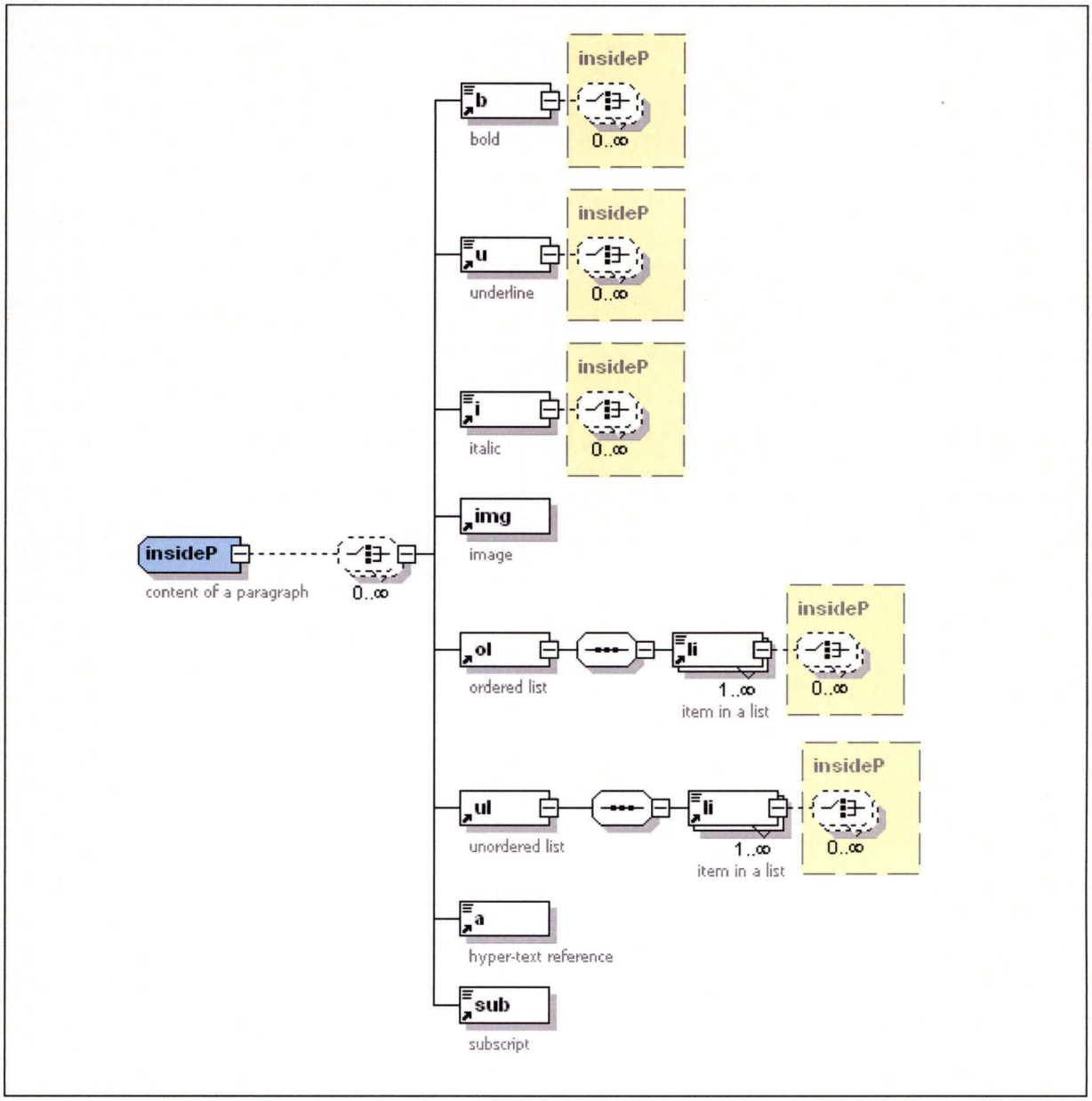


Figure 1.23 ***Type « InsideP »***

1.1.1 Glossaire

Un glossaire est structuré comme suit:

- Titre (glossaire)
- Définitions de tous les mots clés (voir « *Mot clé* »)

Le glossaire contient toutes les définitions des mots clés.

Les définitions sont reprises du glossaire du Manuel COSMIC-FFP.

Au point de vue programmation, un lien est mis vers le glossaire afin d'éviter la redondance. Le glossaire n'est pas un concept propre à CosmicXpert mais simplement un moyen de regrouper tous les mots clés existants afin d'éviter les redondances inutiles.

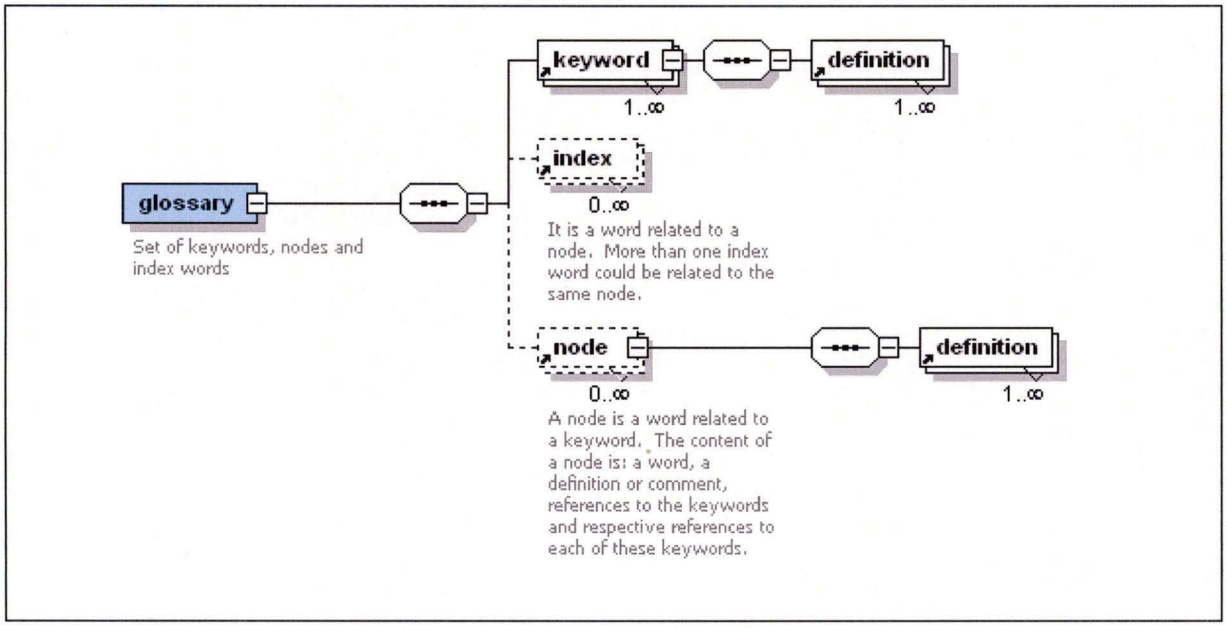


Figure 1.24 ***Contenu du fichier glossary.xml***

1.1.2 Mot-clé et Noeud

Aussi bien les mots-clés que les nœuds sont structurés de la façon suivante :

- Titre (nom du mot-clé ou du noeud)
- Définition du mot-clé ou du noeud

Le mot-clé contient sa définition. Cette définition est reprise du glossaire du Manuel COSMIC-FFP.

Ces différents champs sont illustrés à la Figure 1.24 .

S'il n'y a pas de définition dans le glossaire correspondant à ce mot clé, on doit faire une analogie avec d'autre source (Swebok) et on spécifie « *Not Defined in COSMIC-FFP* ».

La structure de mise en page respectée est celle du Manuel COSMIC-FFP.

Un nœud est un mot référençant un mot-clé. Les relations entre les mots-clés et les nœuds sont définies dans le fichier « *searching.xml* » dont le contenu est représenté à la Figure 1.25.

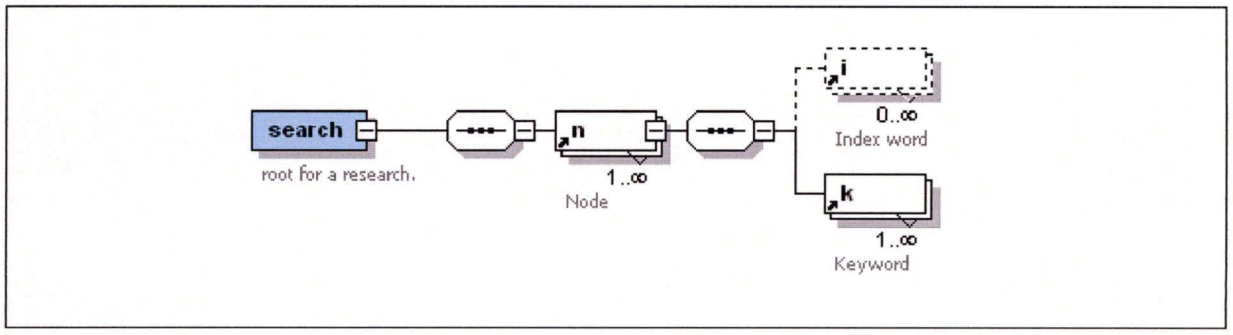


Figure 1.25 ***Contenu du fichier searching.xml***

1.1.3 Mot d'index

Mot d'index **:** Un mot d'index est un mot associé à un nœud. Il peut exister plusieurs mots d'index référençant le même nœud. La relation entre un mot d'index et un nœud ne contient pas de pourcentage.

Un mot d'index est défini comme suit :

- Titre (nom du mot d'index)

Ce champ est illustré à la Figure 1.24 (par le champ « *node* »).

Champ lexical **:** Un champ lexical est un ensemble des mots d'index et des nœuds référençant le même mot clé.

Les relations entre les nœuds et les index sont également définies dans le fichier « *searching.xml* » (Figure 1.25). Les index introduits par un mesureur et n'ayant pas de relations avec un mot-clé sont enregistrés dans le fichier « *uIndexList.xml* » dont le contenu est représenté à la Figure 1.26 :

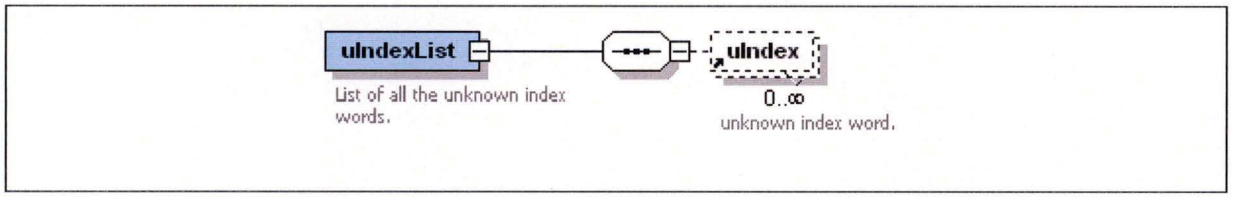


Figure 1.26 ***Contenu du fichier uIndexList.xml***

1.1.4 Concept topologique

Un concept topologique est structuré comme :

- Titre (nom du concept topologique)
- Définition du concept topologique (définition du mot clé le représentant)
- Explication. Ce champ est optionnel.
- Aussi (mots clés liés au concept topologique). Ce champ est optionnel.
- Définitions de mots clés liés à la définition du concept topologique

La définition et l'explication du concept topologique sont reprises du glossaire du Manuel COSMIC-FFP.

Les définitions (1-N) liées au concept topologique sont formées par des mots-clés. Elles sont reprises du glossaire du Manuel COSMIC-FFP ou d'une autres sources (Swebok).

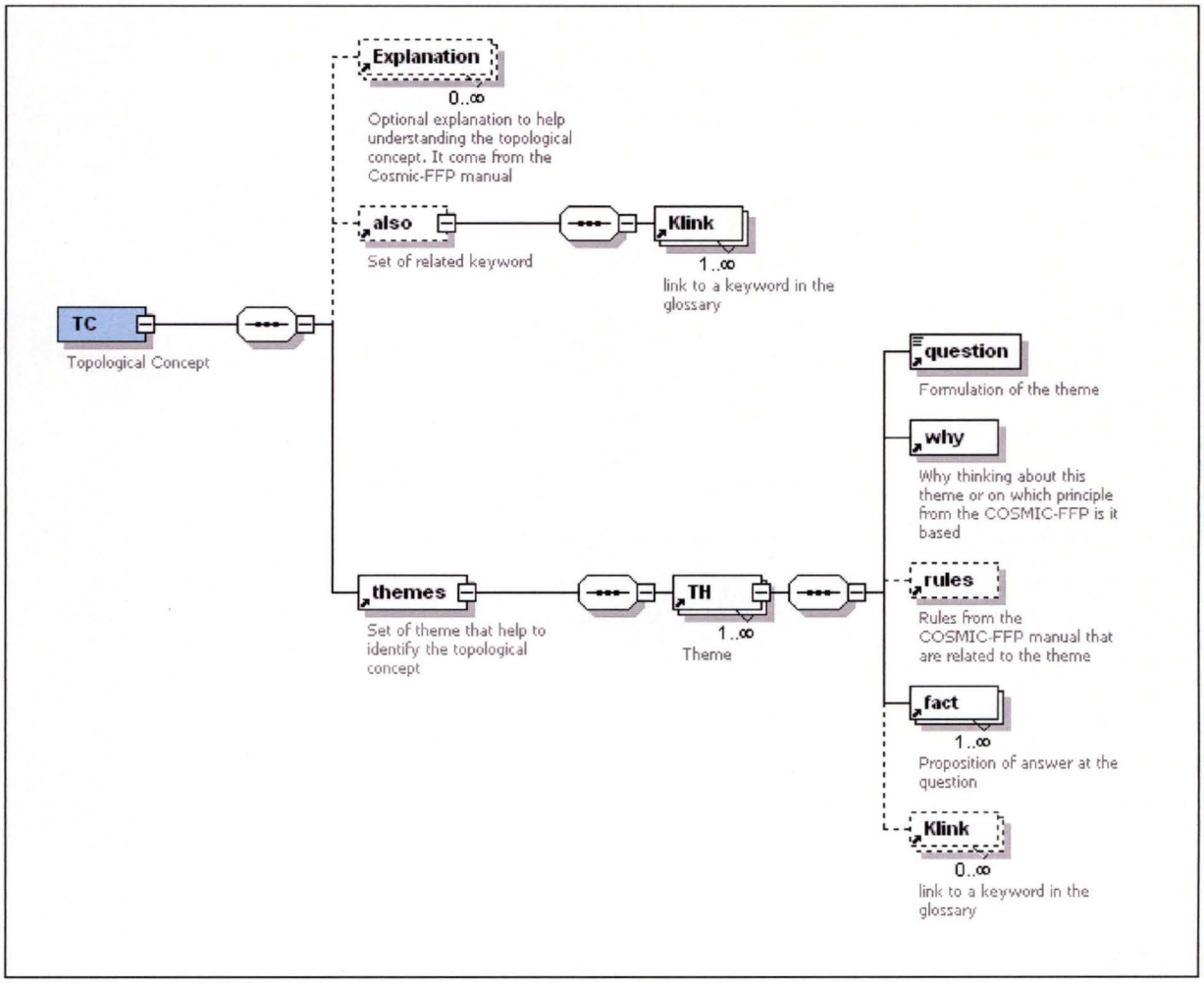


Figure 1.27 ***Contenu d'un fichier d'un concept topologique***

Les relations, ainsi que leurs magnitudes, entre les mots-clés (ou les nœuds) et les concepts topologiques sont enregistrées dans le fichier « *xpert.xml* » dont le contenu est représenté à la Figure 1.28.

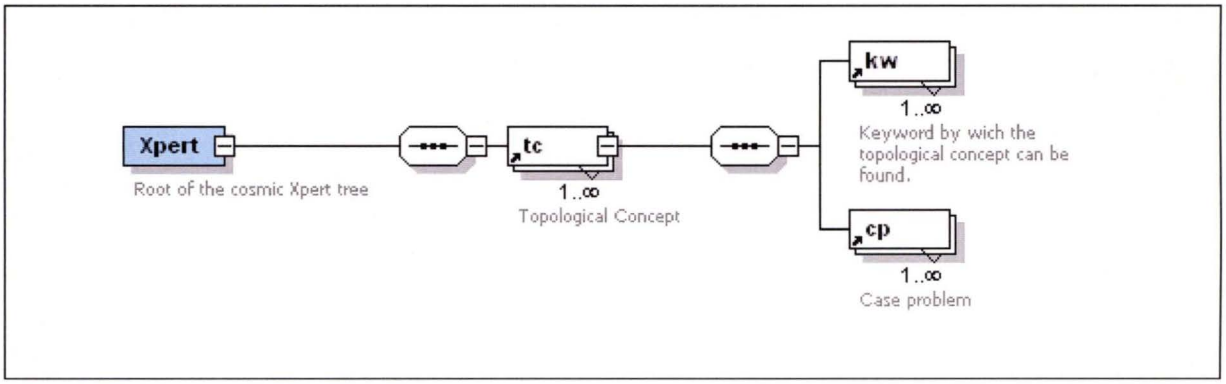


Figure 1.28 ***Contenu du fichier xpert.xml***

1.1.5 Cas d'étude

Un cas d'étude est structuré comme suit :

- Titre (nom du cas problème)
- Résumé
- Contexte (en fonction d'un concept topologique)

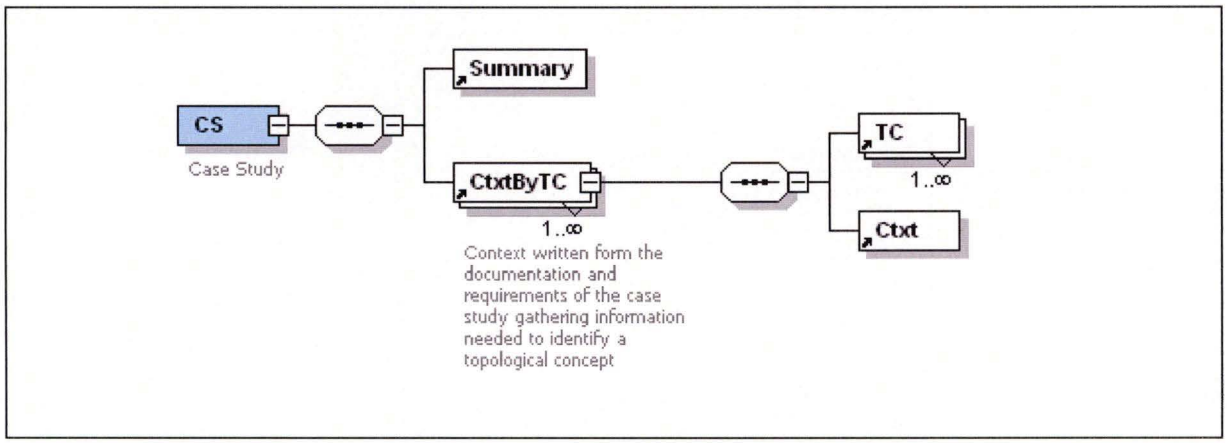


Figure 1.29 ***Contenu d'un fichier d'un cas d'étude***

1.1.6 Cas problème

Un cas problème est structuré comme suit :

- Titre (nom du cas problème)
- Explication (problématique – information sur le cas)
- Contexte
- Définitions des mots clés liés au cas problème.

L'explication du cas problème contient les informations suivantes :

- La problématique contenant une explication de l'objectif de ce cas problème, c'est-à-dire une brève explication de la problématique du cas problème.
- Les exigences propres à ce cas problème. Ces exigences sont extraites de la documentation du cas étudié ou du Manuel COSMIC-FFP pour les cas problèmes génériques.

Le contexte du cas problème contient les informations suivantes :

- Une vue globale du cas problème dans l'application étudiée, suivant un concept topologique donné.
- Les exigences en général (documentation du cas d'étude).

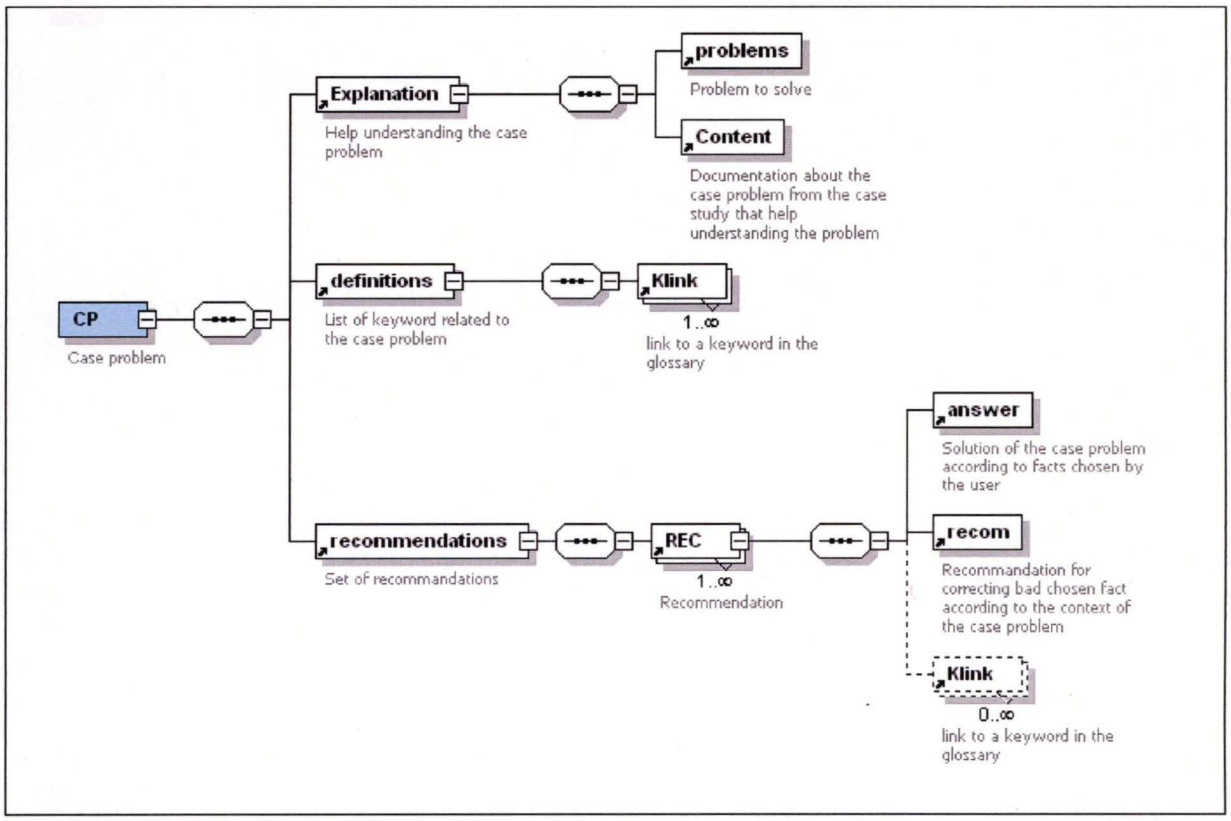


Figure 1.30 ***Contenu d'un fichier d'un cas problème***

1.1.7 Thème

Un thème est structuré comme suit :

- Titre (Question)
- Pourquoi ce thème?
- Contexte
- Règles. Ce champ est optionnel
- Définitions

« *Pourquoi ce thème?* » explique l'intérêt du thème en fonction des principes du Manuel COSMIC-FFP lié à un concept topologique.

Le contexte du thème contient l'explication du cas problème
Les règles sont celles de la méthode COSMIC-FFP qui sont appliquées par l'intermédiaire du thème. Elles sont reprises du Manuel COSMIC-FFP.

Les définitions permettent la compréhension du thème et doivent provenir du glossaire de CosmicXpert.

Ces différents champs sont illustrés à la Figure 1.27 .

1.1.8 Faits

Un fait est structuré comme suit :

- Titre (Réponse)
- Magnitude de la réponse

Ces différents champs sont illustrés à la Figure 1.27 .

1.1.9 Recommandation

Une recommandation est structuré comme suit :

- Réponse (suivant les faits choisis dans CosmicXpert)
- Recommandation (véritable réponse – suggestion)
- Définitions

La réponse contient l'information suivante :

- Quelle est la solution trouvée en ayant répondu aux différents thèmes proposés?

La recommandation contient les informations suivantes :

- la véritable réponse (si on se trouve dans un cas d'étude) ou une réponse suivant les réponses des questions (si on se trouve dans le cas générique),
- des informations d'aide sur l'erreur effectuée,
- des informations sur le thème. C'est-à-dire comment aller chercher l'information nécessaire pour répondre au thème erroné.

Ces différents champs sont illustrés à la Figure 1.30 .

Les définitions sont constituées par des mots clés que l'on retrouve dans le glossaire du Manuel COSMIC-FFP.

1.1.10 Utilisateurs

Afin de contrôler l'accès au système CosmicXpert, il est nécessaire de disposer d'une liste des utilisateurs. Cette liste est enregistrée dans le fichier « *users.xml* » dont le contenu est représenté à la Figure 1.31 :

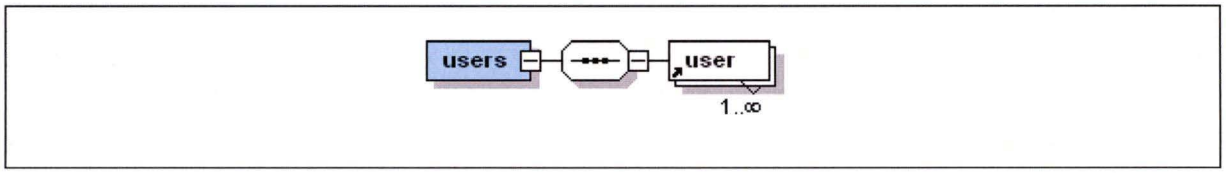


Figure 1.31 ***Contenu du fichier users.xml***

Un utilisateur est structuré comme suit :

- Login (nom)
- Mot de passe
- Statut
- Date d'expiration

1.2 Les différents fichiers XML et leur nomination?

Remarque :

Ne pas laisser d'espace blanc dans les titres (les remplacer par « _ »)

Utiliser les abréviations se trouvant ci-dessous

Tableau 1.2 ***Abréviation des concepts topologique***

Nom du concept	Abréviation
Boundary	B
Data Group	DG
Data Movement	DM
Entry	E
Exit	X
Functional Process	FP
Read	R
Triggering Event	TE
User	U
Write	W

Cette liste n'est pas exhaustive et peut évoluer en fonction de la norme COSMIC-FFP.

Les suffixes _MIS et _RT peuvent être ajoutés pour déterminer des concepts propres aux applications de gestion (MIS) ou en temps réel (RT).

Attention, ces abréviations peuvent uniquement être utilisées par facilité dans la nomination des fichiers et pas dans le contenu des fichiers.

Tableau 1.3 *Abréviations des cas d'étude*

Nom du cas étudié	Abréviation
Data Warehouse	W
Report Generator	RG
Rice Cooker	RC
Valve Control	V
Cas Générique	G

1.2.1 Concept

[e_][nom complet du concept de CosmicXpert][.xml]
Exemple : « e_mot-clé.xml »

1.2.2 Concept topologique

[etc_][abréviation du concept topologique][.xml]
Exemple : « etc_DG_MIS.xml »

1.2.3 Cas d'étude

[nom complet du cas d'étude][.xml]
Exemple : « riceCooker.xml »

1.2.4 Cas problème

[ecp_][abréviation du concept topologique][_][nom du cas problème][_][abréviation du cas d'étude][.xml]
Exemple : « ecp_DG_customer_entity_W.xml »

Les informations concernant les thèmes et les faits se retrouvant dans le fichier de leur concept topologique parent, il n'existe pas de fichiers de ces concepts.

Les informations concernant tous les mots-clés, noeuds et index se retrouvent dans le fichier glossary.xml.

Il est à noter que les mots d'index introduits par les mesureurs mais n'ayant pas de correspondance avec un mot-clé, se trouvent dans le fichier uIndexList.xml.

1.3 Comment nommer les images et comment les lier dans le fichier XML?

Remarque :
Ne pas laisser d'espace blanc dans les titres (les remplacer par « _ »)
Le numéro de l'image se trouvant avant l'extension « .gif » est le numéro de l'emplacement de l'image dans le fichier XML, c'est-à-dire la première image a partir du début du document prendra toujours le numéro 1 et ainsi de suite.

1.3.1 Concept

[nom complet du concept de CosmicXpert_numéro de l'image][.gif]
Exemple : « mot-clé_1.gif »

1.3.2 Mot clé et Noeud

[nom complet du mot clé][_][numéro de l'image][.gif]
Exemple : « Data_Group_1.gif »

1.3.3 Concept topologique

[etc_][abréviation du concept topologique][_][numéro de l'image][.gif]
Exemple : « etc_DG_MIS_1.gif »

1.3.4 Cas problème

[ecp_][abréviation du concept topologique][_][nom du cas problème][_][abréviation du cas d'étude][_][numéro de l'image][.gif]
Exemple : « ecp_DG_customer_entity_W_1.xml »

1.3.5 Theme

[th_][abréviation du concept topologique][_][nom du cas problème][_][nom de la question][_][abréviation du cas d'étude][_][numéro de l'image][.gif]
Exemple : « eth_DG_customer_entity_identification_of_attribute_W_1.gif »

1.3.6 Recommendation

[rec_][nom du concept topologique][_][nom du cas problème][_][nom de la recommandation][_][abréviation du cas d'étude][_][numéro de l'image][.gif]
Exemple : « rec_DG_customer_entity_identify_W_1.gif »

1.4 Comment nommer les fichiers Xsd et comment les lier dans le fichier XML?

Remarque :
Ne pas laisser d'espace blanc dans les titres (les remplacer par « _ »)

Les fichiers XSD existant dans le répertoire Xsd sont

- cases.xsd
- Concept.xsd
- cp.xsd
- csList.xsd
- global.xsd
- glossary.xsd
- searching.xsd
- tc.xsd
- testReport.xsd
- uIndexList.xsd
- users.xsd
- xpert.xsd

Le premier élément d'un fichier XML doit contenir le lien vers son schéma XML. Liaison se fait en ajoutant les attributs suivants dans la balise racine de ce fichier:

« xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance

xsi:noNamespaceSchemaLocation="../xsd/le_fichier.xsd" »

1.5 Comment nommer les fichiers Xsl et comment les lier dans le fichier XML?

Remarque :

Ne pas laisser d'espace blanc dans les titres (les remplacer par « _ »)

Les fichiers Xsl utilisés pour la transformation du fichier XML en HTML se trouvent dans le répertoire Xsl.

Pour les concepts de base, les fichiers existant sont :

- Case.xsl
- Concept.xsl
- glossary.xsl
- tc.xsl
- cp.xsl
- theme.xsl
- rec.xsl

Pour ces fichiers, la ligne : « <?xml-stylesheet type="text/xsl" href=" ../xsl/le_fichier.xsl"?> » doit être présente au début du fichier pour le lier avec son transformateur.

1.6 Comment stocker les fichiers XML - GIF – XSD – XSLT dans les dossiers?

Remarque : les nom se trouvant entre « » sont les noms des dossiers.

Un dossier «*reference*» contiendra toutes les références de la base de connaissance comme :

- «Xsl» contient tous les fichiers xsl utilisés pour l'affichage des fichiers xml.
- «Xsd» contient tous les fichiers xsd utilisés pour valider les fichiers xml.
- «*C*» contient tous les concept propre à CosmicXpert.
- «*Kw*» contient tous les mots clés
- «*Tc*» contient tous les Concept topologique
- «*Cp*» contient tous les Cas problème
- «*Th*» contient tous les thèmes
- «*Rec*» contient toutes les recommandations
- «*Cs*» contient toutes les documentations des cas d'étude

Le dossier des cas problèmes, à savoir le dossier « ./CP » comporte les dossiers de tous les cas d'étude :

- un dossier «*G*» pour le cas d'étude « *Cas Générique* »
- un dossier «*Rc*» pour le cas d'étude « *Rice Cooker* »
- un dossier «*W*» pour le cas d'étude « *Data Warehouse* »
- un dossier «*V*» pour le cas d'étude « *Valve Control* »
- un dossier «*Rg*» pour le cas d'étude « *Report Generator* »

Chacun de ces dossiers contient les fichiers xml des cas problèmes correspondant au cas étudié (voir comment nommer les fichiers xml).

Afin de faciliter la compréhension, la reprend une représentation des liens entre les différents concepts de CosmicXpert et de COSMIC-FFP.

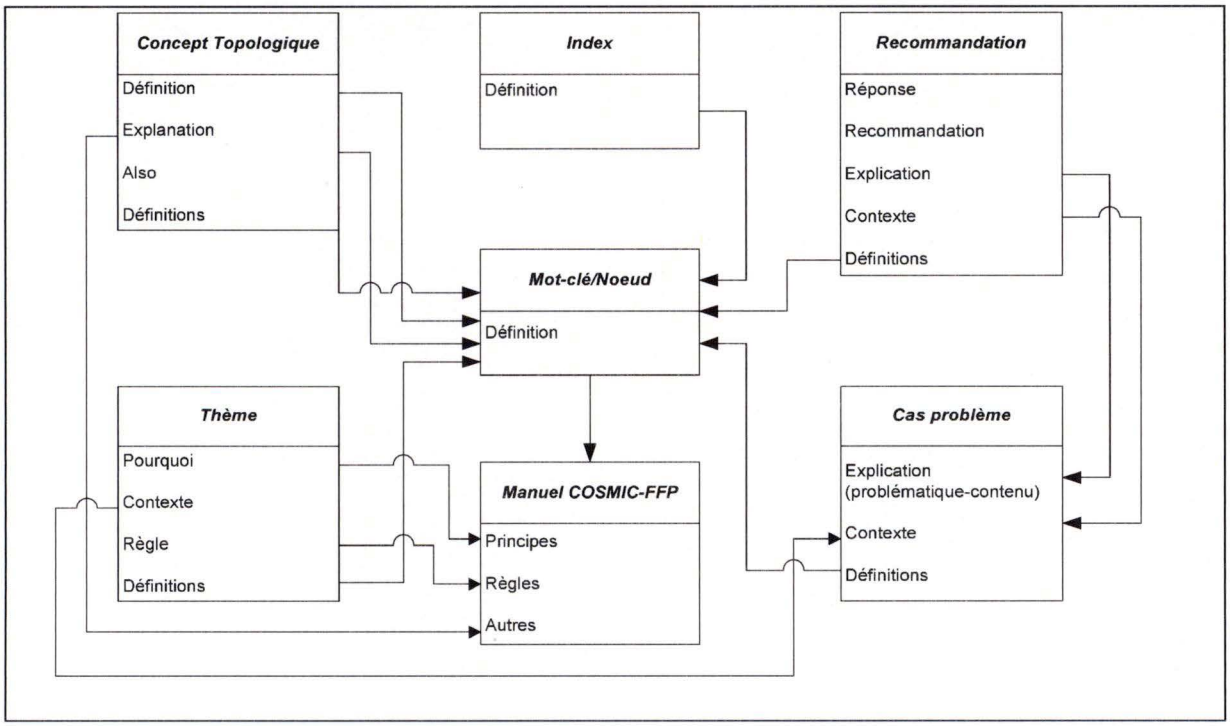


Figure 1.32 *liens entre les différents concepts de CosmicXpert et de COSMIC-FFP*

Annexe 2 : Cas d'utilisations de l'interface expert

Use Case: Start a session
ID: UC 1
Acteurs : Mesureur Expert Administrateur
Préconditions : 1. L'acteur a démarré le système, i.e. a ouvert la page Web de CosmicXpert.
Scénario Normal : 1. Le cas d'utilisation commence quand l'acteur ouvre la page Web de CosmicXpert. 2. L'acteur introduit un nom d'utilisateur et un mot de passé valides 3. Le système vérifie la date d'expiration de l'utilisateur. 4. Le système affiche l'interface correspondante au statut de l'acteur
Scénarios Alternatifs : InvalidUsername ou InvalidPassword DateExpire
Postconditions : 1. L'acteur a accès au système.

Use Case: Start a session ***Scénario Alternatif : InvalidUsername or Invalid Password***
ID: UC 1.1
Acteurs : Mesureur Expert Administrateur
Préconditions :
Scénario Alternatif : 1. Le cas d'utilisation commence à l'étape 2 du cas *Start a session* un nom d'utilisateur ou un mot de passé invalide. 2. Le système informe l'acteur qu'il a introduit de mauvaises valeurs et lui demande d'introduire son nom d'utilisateur et son mot de passé à nouveau.
Postconditions :

Use Case: Start a session ***Scénario Alternatif : DateExpires***
ID: UC 1.2
Acteurs : Mesureur Expert Administrateur
Préconditions :
Scénario Alternatif : 1. Le cas d'utilisation commence à l'étape 3 du cas *Start a Session* quand le système vérifie la date d'expiration de l'utilisateur, et que cette date est dépassée. 2. Le système informe l'acteur que sa date d'utilisation a expiré et qu'il n'est plus autorisé à utiliser le système.
Postconditions :

Use Case: Add a Keyword
ID: UC 2.1
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert sélectionne "Add a Keyword" dans l'interface du système. 2. Le système demande à l'expert d'introduire le nom du nouveau mot-clé, sa description, les concepts topologiques auxquels il sera relié, ainsi que le pourcentage correspondant à la magnitude de ce lien. 3. L'expert introduit ces valeurs. 4. Si le mot-clé existe déjà 4.1. Le système informe l'expert que le mot-clé existe déjà. 4.2. Le cas d'utilisation se termine. 5. Sinon 5.1. Le système ajoute le nouveau mot-clé.
Postconditions : La base de connaissances est mise à jour.

Use Case: Modify a Keyword
ID: UC 2.2
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "modify" quand un mot-clé est sélectionné. 2. L'expert modifie le nom et/ou la description et/ou le pourcentage de la relation avec les concepts topologiques choisis. 3. Le système appliqué les changements au mot-clé sélectionné.
Postconditions : La base de connaissances est mise à jour.

Use Case: Delete a Keyword
ID: UC 2.3
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "delete" quand un mot-clé est sélectionné. 2. Le système affiche la liste des concepts topologiques auxquels est relié ce mot-clé. 3. Le système demande à l'expert une confirmation de suppression du mot-clé. 4. Si l'expert confirme la suppresion 4.1. Si tous les concepts topologiques auxquels sont rattachés ce mot-clé sont lies avec au moins un autre mot-clé 4.1.1. Le système supprime le mot-clé. 4.2. Sinon 4.2.1. Le système informe l'expert qu'il n'est pas autorisé à supprimer ce mot-clé. 4.2.2. Le cas d'utilisation se termine. 5. Sinon 5.1. Le cas d'utilisation se termine.
Postconditions : La base de connaissances est mise à jour.

Use Case: Add a Topological Concept
ID: UC 3.1
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "Add a Topological Concept". 2. Le système demande à l'expert d'introduire le nom du nouveau mot-clé, correspondent à la definition du nouveau concept topologique. (*Add a keyword*) 3. Le système demande à l'expert d'introduire le nom du nouveau concept toplogique, sa description, les mot-clé (et/ou noeud) associées et les pourcentages correspondant à ces relations. 4. Le système demande à l'expert d'introduire de nouveaux themes associés (*Add a theme*). 5. Le système demande à l'expert d'introduire de nouveaux cas problèmes (*Add a case problem*) et leurs recommandations correspondantes (*Add a recommendation*) 6. L'expert introduit ces valeurs. 7. Si le concept topologique existe déjà 7.1. Le système informe l'expert que le concept topologique existe déjà. 7.2. Le cas d'utilisation se termine. 8. Sinon 8.1. Le système ajoute le nouveau concept topologique et tous les concepts associés.
Postconditions : La base de connaissances est mise à jour.

Use Case: Modify a Topological Concept
ID: UC 3.2
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "modify" quand un concept topologique est sélectionné. 2. L'expert modifie le nom et/ou la description et/ou les relations avec les mots-clés (et/ou noeuds) ainsi que leurs pourcentages associées. 3. Le système applique les changements au concept topologique sélectionné.
Postconditions : La base de connaissances est mise à jour.

Use Case: Add a Case Problem
ID: UC 4.1
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "add" quand un concept topologique est sélectionné. 2. Le système demande à l'expert d'introduire le nom du nouveau cas problème, sa description, les concepts topologiques associées ainsi que les pourcentages de ces relations. 3. L'expert introduit ces valeurs. 4. Si le cas problème existe déjà 4.1. Le système informe l'expert que le cas problème existe déjà. 4.2. Le cas d'utilisation se termine. 5. Sinon 5.1. Le système ajoute le nouveau cas problème.
Postconditions : La base de connaissances est mise à jour.

Use Case: Modify a Case Problem
ID: UC 4.2
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "modify" quand un cas problème est sélectionné. 2. L'expert modifie le nom et/ou la description et/ou les relations avec le concept topologique sélectionné ainsi que le pourcentage. 3. Le système applique les changements au cas problème sélectionné.
Postconditions : La base de connaissances est mise à jour.

Use Case: Delete a Case Problem
ID: UC 4.3
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "delete" quand un cas problème est sélectionné. 2. Le système affiche la liste des recommandations associées à ce cas problème. 3. Le système demande une confirmation à l'expert et l'informe que s'il poursuit les recommandations seront aussi supprimées. 4. Si l'expert confirme la suppression 4.1. Si ce cas problème est le dernier cas associé à un concept topologique 4.1.1. Le système informe l'expert que le cas problème ne peut pas être supprimé. 4.1.2. Le cas d'utilisation se termine. 4.2. Sinon 4.2.1. Le système supprime le cas problème et toutes les recommandations associées. 5. Sinon 5.1. Le cas d'utilisation se termine.
Postconditions : La base de connaissances est mise à jour.

Use Case: Add a Theme
ID: UC 5.1
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "add" quand un concept topologique est sélectionné. 2. Le système demande à l'expert d'introduire le nom du nouveau thème, sa description et le pourcentage des associations avec le concept topologique sélectionné. 3. L'expert introduit ces valeurs. 4. Si ce thème existe déjà pour ce concept topologique 4.1. Le système informe l'expert que le thème existe déjà. 4.2. Le cas d'utilisation se termine. 5. Sinon 5.1. Le système ajoute le nouveau thème.
Postconditions : La base de connaissances est mise à jour.

Use Case: Modify a Theme
ID: UC 5.2
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "modify" quand un thème est sélectionné. 2. L'expert change le nom et/ou la description et/ou le pourcentage des relations avec les concepts topologiques choisis. 3. Le système applique les changements au thème sélectionné.
Postconditions : La base de connaissances est mise à jour.

Use Case: Delete a Theme
ID: UC 5.3
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "delete" quand un thème est sélectionné. 2. Le système affiche la liste des faits associés à ce thème. 3. Le système demande à l'expert de confirmer la suppression. 4. Si l'expert confirme la suppression du thème 4.1. Si le thème est le dernier thème associé à un concept topologique 4.1.1. Le système informe l'expert que le theme ne peut pas être supprimé. 4.1.2. Le cas d'utilisation se termine. 4.2. Sinon 4.2.1. Le système supprime le thème. 5. Sinon 5.1. Le cas d'utilisation se termine.
Postconditions : La base de connaissances est mise à jour.

Use Case: Add a Recommendation
ID: UC 6.1
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "add" quand un cas problème est sélectionné. 2. Le système demande à l'expert d'introduire le nom de la nouvelle recommandation, sa description et les pourcentages minimum et maximum de sa portée. 3. L'expert introduit ces valeurs. 4. Si la recommandation existe déjà 4.1. Le système informe l'expert que la recommandation existe déjà. 4.2. Le cas d'utilisation se termine. 5. Sinon 5.1. Le système ajoute la nouvelle recommandation.
Postconditions : La base de connaissances est mise à jour.

Use Case: Modify a Recommendation
ID: UC 6.2
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "modify" quand une recommandation est sélectionnée. 2. L'expert modifie le nom et/ou la description et/ou les pourcentages minimum et maximum de la portée. 3. Le système applique les changements à la recommandation.
Postconditions : La base de connaissances est mise à jour.

Use Case: Delete a Recommendation
ID: UC 6.3
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "delete" quand une recommandation est sélectionnée. 2. Le système affiche les modifications qui seront apportées aux autres recommandations du groupe. 3. Le système demande à l'expert de confirmer la suppression. 4. Si l'expert confirme la suppression de la recommandation 4.1. Si le groupe de recommandations associées à ce cas problème ne contient plus que 2 recommandations 4.1.1. Le système informe l'expert que la recommandation ne peut pas être supprimée. 4.1.2. Le cas d'utilisation se termine. 4.2. Sinon 4.2.1. Le système supprime la recommandation. 5. Sinon 5.1. Le cas d'utilisation se termine.
Postconditions : La base de connaissances est mise à jour.

Use Case: Add a Fact
ID: UC 7.1
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "add" quand un thème est sélectionné. 2. Le système demande à l'expert d'introduire le nom du fait, sa description, ainsi que le pourcentage de la relation avec le theme sélectionné. 3. L'expert introduit ces valeurs. 4. Si le fait existe déjà pour ce thème 4.1. Le système informe l'expert que le fait existe déjà. 4.2. Le cas d'utilisation se termine. 5. Sinon 5.1. Le système ajoute le nouveau fait.
Postconditions : La base de connaissances est mise à jour.

Use Case: Modify a Fact
ID: UC 7.2
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "modify" quand un fait est sélectionné. 2. L'expert modifie le nom et/ou la description et/ou le pourcentage du fait sélectionné. 3. Le système applique les changements au fait sélectionné.
Postconditions : La base de connaissances est mise à jour.

Use Case: Delete a Fact
ID: UC 7.3
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "delete" quand un fait est sélectionné. 2. Le système demande une confirmation à l'expert. 3. Si l'expert confirme la suppression du fait 3.1. Si le fait est le dernier fait associé à un thème 3.1.1. Le système informe l'expert que le fiat ne peut pas être supprimé. 3.1.2. Le cas d'utilisation se termine. 3.2. Sinon 3.2.1. Le système supprime le fait. 4. Sinon 4.1. Le cas d'utilisation se termine.
Postconditions : La base de connaissances est mise à jour.

Use Case: Switch to Measurer Interface
ID: UC 8
Acteurs : Expert
Préconditions : 1. L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "Measurer Interface" 2. Le système affiche la fenêtre du mesureur.
Postconditions : 1. L'expert peut utiliser toutes les fonctionnalités accessibles à partir de l'interface du mesureur.

Use Case: Consult Searched Words
ID: UC 9.1
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit de consulter les mots recherchés, i.e choisit « Consult Searched Words ». 2. Le système affiche la liste des mots recherchés par les mesureurs qui n'ont pas aboutis à un mot-clé.
Postconditions :

Use Case: Add a Node
ID: UC 9.2
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "Add a new Node". 2. Le système demande à l'expert d'introduire le nom du nouveau noeud, sa description, et le pourcentage des relations avec les mots-clés sélectionnés. 3. L'expert introduit ces valeurs. 4. Si le noeud existe déjà 4.1. Le système informe l'expert que le noeud existe déjà. 4.2. Le cas d'utilisation se termine. 5. Sinon 5.1. Le système ajoute le noeud.
Postconditions : La base de connaissances est mise à jour.

Use Case: Add an Index
ID: UC 9.3
Acteurs : Expert
Préconditions : L'acteur a accès au système.
Flux des événements : 1. Le cas d'utilisation commence quand l'expert choisit "Add a new Index". 2. Le système demande à l'expert d'introduire le nom du nouvel index. L'expert peut introduire un ou plusieurs index. 3. Si l'index existe déjà pour le noeud sélectionné 3.1. Le système informe l'expert que l'index existe déjà pour ce noeud. 3.2. Le cas d'utilisation se termine. 4. Sinon 4.1. Le système ajoute l'index.
Postconditions : La base de connaissances est mise à jour.

Annexe 3 : Page d'administration

La page d'administration sert essentiellement à une gestion des utilisateurs. Elle permet de :

- créer de nouveaux utilisateurs,
- changer le statut des utilisateurs,
- changer la date d'expiration des utilisateurs,
- changer le mot de passe utilisateurs,
- supprimer des utilisateurs.

En plus de ces fonctions, elle permet également de remettre la base de connaissance en état dans le cas où un incident grave ce serait produit sur celle-ci.

3.1 Fonctions de gestion des utilisateurs

La Figure 3.33 représente la page principale. C'est à partir de celle-ci qu'un administrateur peut effectuer toutes les fonctions qu'il désire sur les utilisateurs.

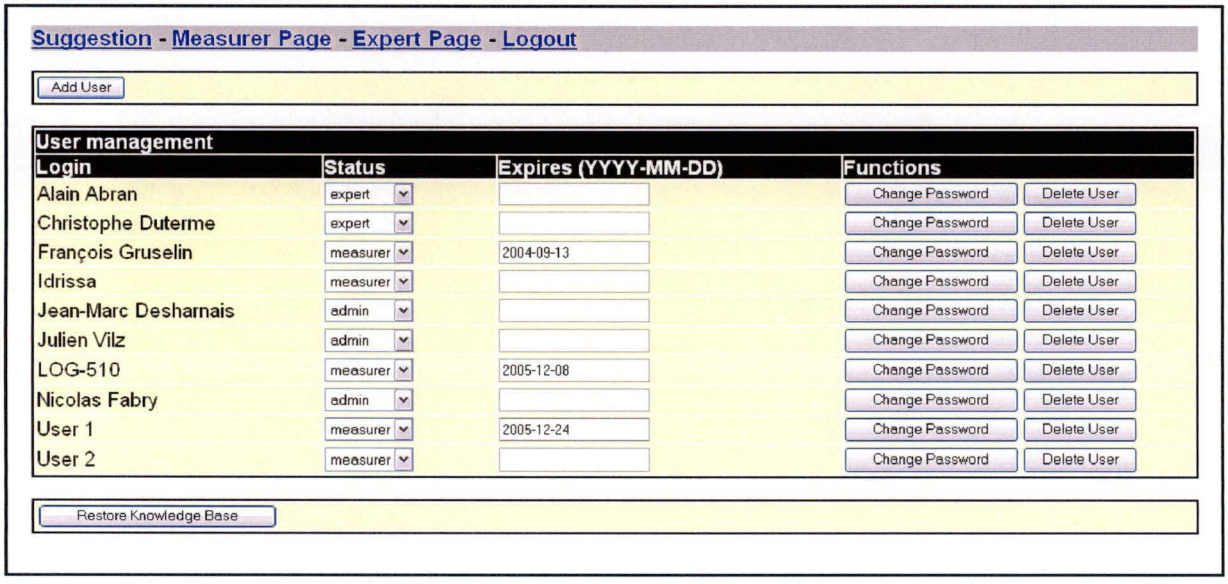


Figure 3.33 *Page d'administration*

3.1.1 Ajout d'un nouvel utilisateur

Pour ajouter un nouvel utilisateur, il suffit à l'administrateur de cliquer sur le bouton « *Add User* » (cf. Figure 3.33). La fenêtre d'ajout apparaît (Figure 3.34). Une fois les différentes informations relatives au nouvel utilisateur introduites, une nouvelle page apparaît confirmant l'ajout de l'utilisateur. Cette dernière page propose aussi l'ajout d'un autre utilisateur.

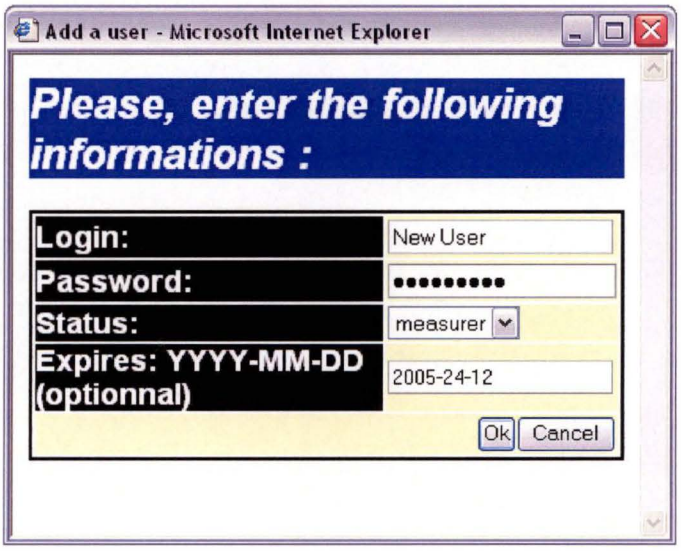


Figure 3.34 *Ajout d'un nouvel utilisateur*

3.1.2 Changement du statut d'un utilisateur

Pour changer le statut d'un utilisateur, il suffit à l'administrateur de sélectionner le nouveau statut de celui-ci dans la liste déroulante située dans la colonne « *Status* » (Figure 3.35). Le statut est changé instantanément.

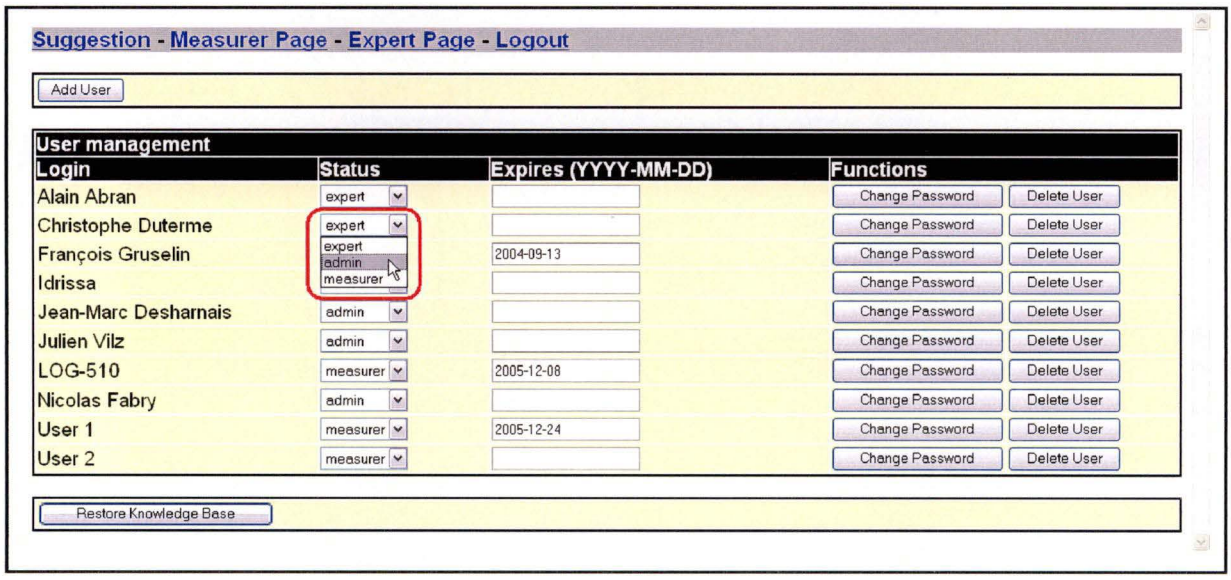


Figure 3.35 *Changement du statut d'un utilisateur*

3.1.3 Changement de la date d'expiration liée à un utilisateur

Pour changer la date d'expiration liée à un utilisateur, il suffit à l'administrateur d'introduire une nouvelle valeur dans la colonne « *Expires* ». Cette nouvelle valeur doit respecter la forme AAAA-MM-JJ (« A » pour « année », « M » pour « mois » et « J » pour « jour »). Dès que l'administrateur clique en dehors du champ « *Expires* », la date d'expiration est immédiatement modifiée.

3.1.4 Changement du mot de passe d'un utilisateur

Pour changer le mot de passe d'un utilisateur, il suffit à l'administrateur de cliquer sur le bouton « *Change Password* » en regard de l'utilisateur en question. Une nouvelle fenêtre apparaît rappelant l'ancien mot de passe et proposant un champs afin que l'administrateur y introduise le nouveau mot de passe ().

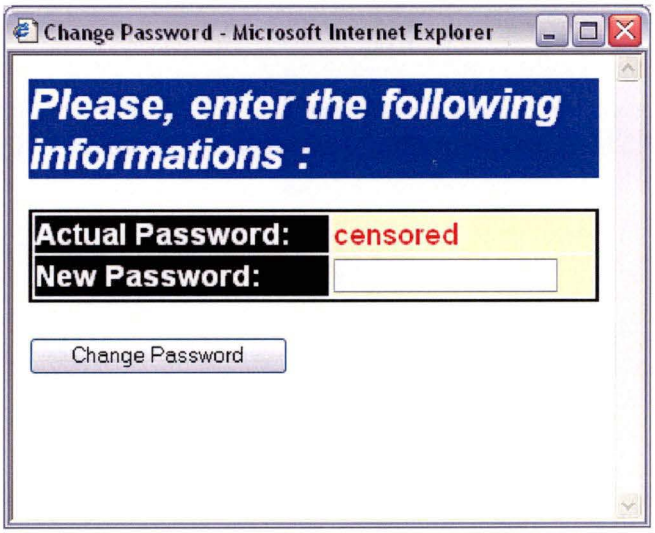


Figure 3.36 *Changement du mot de passe d'un utilisateur*

3.1.5 Suppression d'un utilisateur

Pour supprimer un utilisateur, il suffit à l'administrateur de cliquer sur le bouton « *Delete User* » en regard de l'utilisateur en question. L'utilisateur est alors immédiatement supprimé.

3.2 Fonction de restauration de la base de connaissance

Le bouton « *Restore Knowledge Base* » permet à un administrateur de remettre la base de connaissance en état dans le cas où un incident grave se produit sur celle-ci.

Cette fonction a pour objectif de récupérer la dernière sauvegarde complète de la BC et de réappliquer les dernières versions de chaque fichier ayant été modifiés depuis la prise de cette dernière sauvegarde.

Il est à noter que l'utilisation de cette fonction n'est possible que si les fichiers nécessaires au lancement de l'application n'ont pas été endommagés, sans quoi une récupération est indispensable.

Annexe 4 : Niveau d'indexation

4.1 Le niveau d'indexation dans la page du mesureur

Pour pouvoir utiliser CosmicXpert, et en vue d'identifier des concept, un mesureur doit réaliser la recherche d'un mot-clé. Afin de faciliter cette recherche, une innovation a été apportée dans le prototype 3. Il s'agit d'un niveau d'indexation supplémentaire, composé de deux concepts distincts :

- les mots d'index,
- les *nœuds*.

De façon à permettre la compréhension de la suite de cette annexe, il nous est nécessaires de donner quelques définitions.
Un *champ lexical* est un ensemble de mots d'index et de nœuds référençant le même mot-clé.
Un *mot d'index* est un mot associé à un nœud. Il peut exister plusieurs mots d'index référençant le même nœud. La relation entre un mot d'index et un nœud ne contient pas de pourcentage.
Un *nœud* est un mot référençant un mot-clé. Le nœud est composé d'un mot, d'une définition (commentaire), et de références vers des mots-clés, ainsi que les pourcentages respectifs de ces références.
Un mot recherché est un mot entré par un mesureur dans le champ de texte de recherche d'un mot-clé.

La Figure 4.37 présente le bouton permettant à un mesureur de réaliser une recherche libre vis l'index.
La fenêtre (de type « *pop-up* ») qui lui est affichée est donnée à la Figure 4.38.

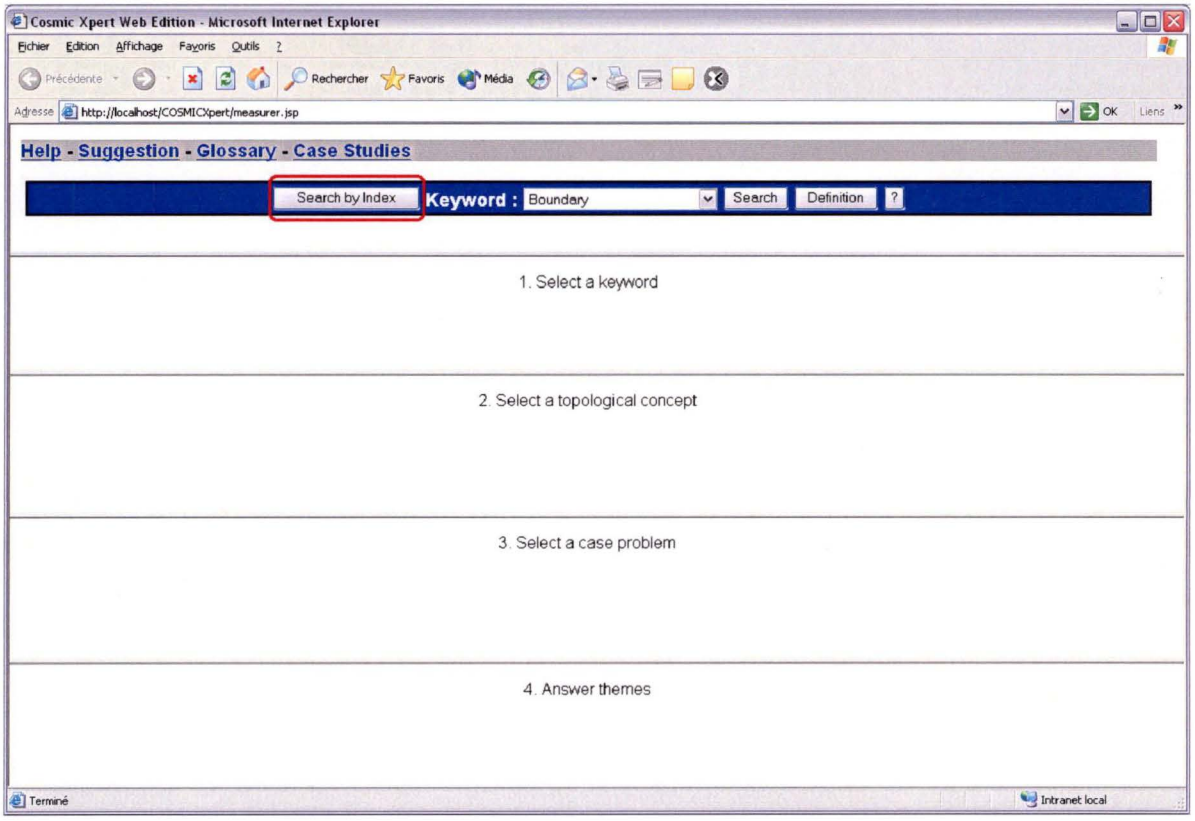


Figure 4.37 *Sélection de la recherche via l'index sur la page du mesureur*

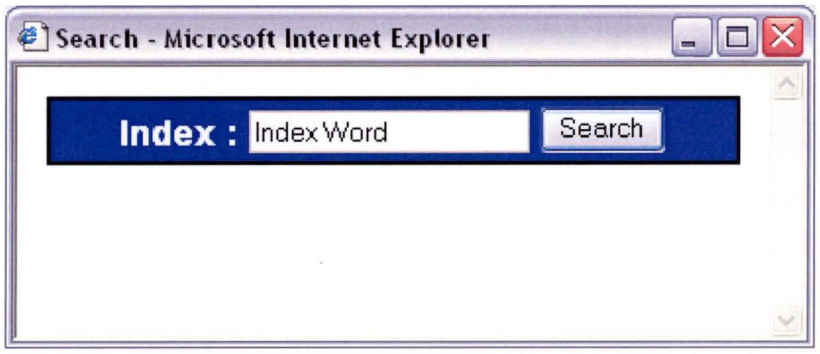


Figure 4.38 *Pop-up d'introduction d'un mot recherché*

Une fois que le mesureur a introduit son mot recherché, le système lui affiche le résultat de la recherche. Si un lien vers un ou plusieurs mots-clés existe, le mesureur pourra sélectionner l'un de ces mots-clés et poursuivre sa recherche, comme le montre la . Sinon, le système l'avertit qu'aucune référence ne correspond à ce mot recherché. Dans ce dernier cas de figure, le système enregistre le mot recherché dans une liste, afin de permettre à des experts de vérifier quels mots ont été recherchés, facilitant la mise à jour du système et l'amélioration de ses performances. Nous examinons ces fonctions plus en détail dans la section suivante.

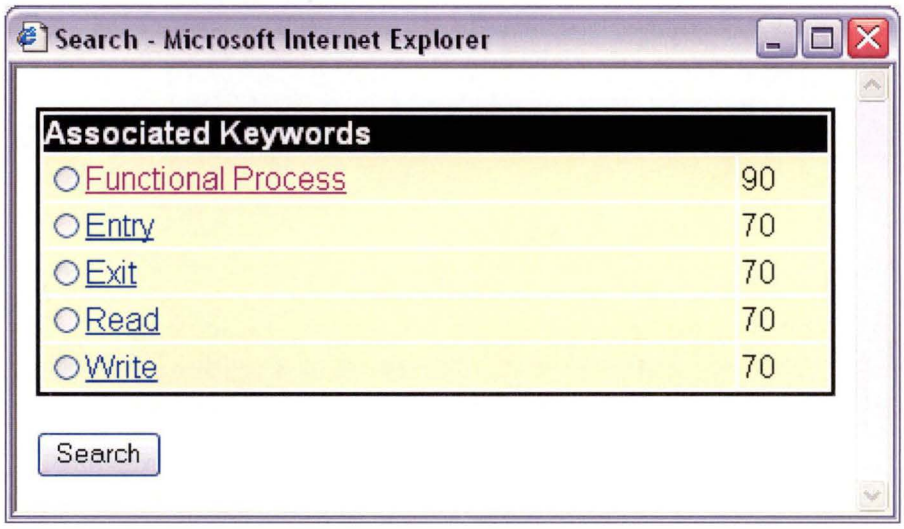


Figure 4.39 *Exemple de Pop-up des résultats de la recherche*

4.2 Le niveau d'indexation pour l'expert

L'ajout de la fonctionnalité de recherche libre entraîne des modifications pour le mesureur mais aussi pour l'expert. En effet, il se voit investi de quelques fonctions supplémentaires.
La Figure 4.40 présente la page de l'expert réservée à la gestion du niveau d'indexation.

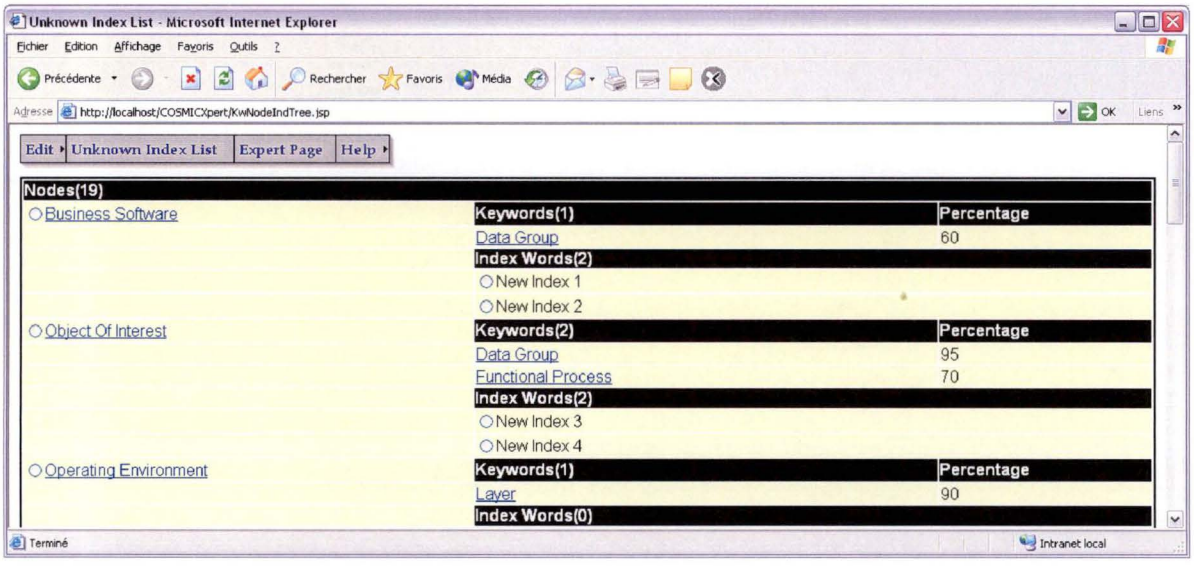


Figure 4.40 *Page de l'expert permettant la gestion du niveau d'indexation*

L'une des nouvelles fonctions attribuées à l'expert, concerne la gestion des mots recherchés n'ayant pas aboutis. Comme mentionné dans la section précédente, si un mesureur introduit

un mot dans le système mais que la recherche est vide pour ce mot, le système mémorise ce mot, afin que les experts puissent avoir une trace de ce que recherche les mesureurs. De cette façon, il leur est possible de créer de nouvelles entrées, améliorant les performances du système de recherche. Cette fonction est accessible à l'expert lorsqu'il choisit « *Unknown Index List* » comme présenté à la Figure 4.40.

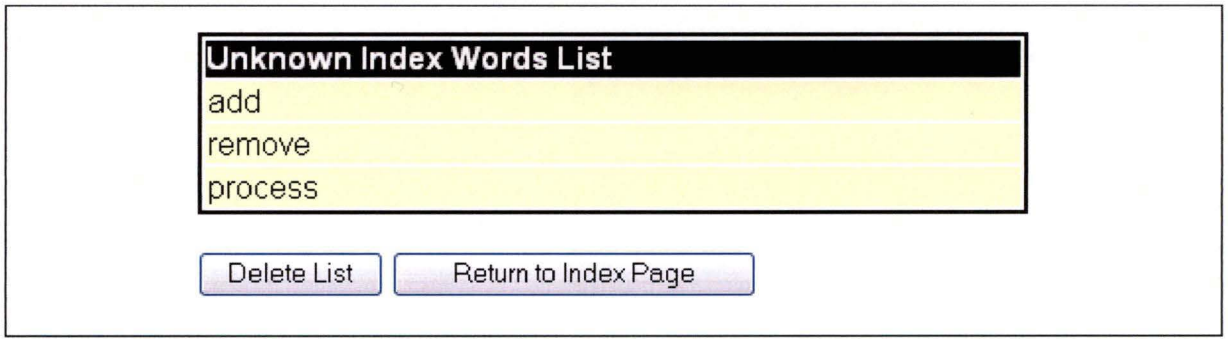


Figure 4.41 *Exemple d'une liste de mots recherchés par des mesureurs, n'ayant pas donnés de résultats*

4.2.1 Introduction d'un mot d'index

Le niveau de recherche le plus simple est le mot d'index. Il est très facile pour un expert de créer des mots d'index, référençant un ou plusieurs nœuds. Pour ce faire, il lui suffit de sélectionner un nœud dans la liste qui lui est affichée sur la page principale de gestion des index, et de sélectionner la fonction « *Add* ». Ensuite, il sera prié d'introduire le nom du nouveau mot d'index qu'il souhaite ajouter. Il a la possibilité d'introduire un plusieurs mots d'index.

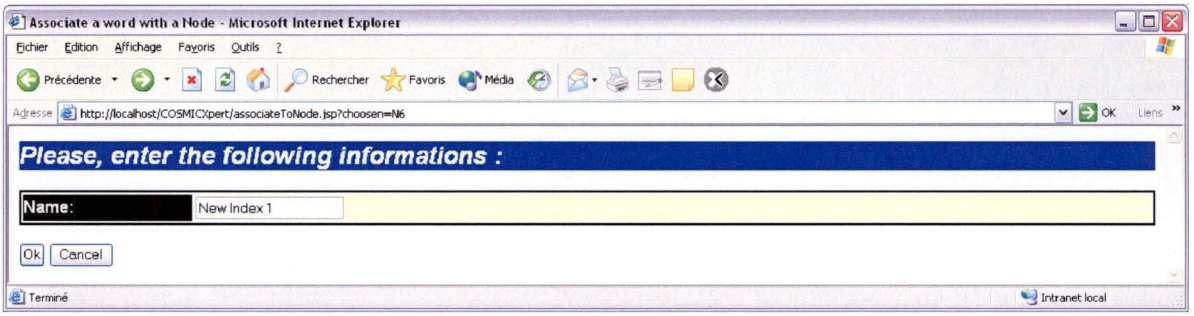


Figure 4.42 *Ajout d'un nouveau mot d'index*

4.2.2 Ajout d'un nœud

Une autre possibilité offerte à l'expert pour ajouter des références, est de créer un nouveau nœud, associé à un ou plusieurs mots-clés. Pour ce faire, il lui suffit de sélectionner la fonction « *Add a node* » sur la page de l'index, comme le montre la Figure 4.43.

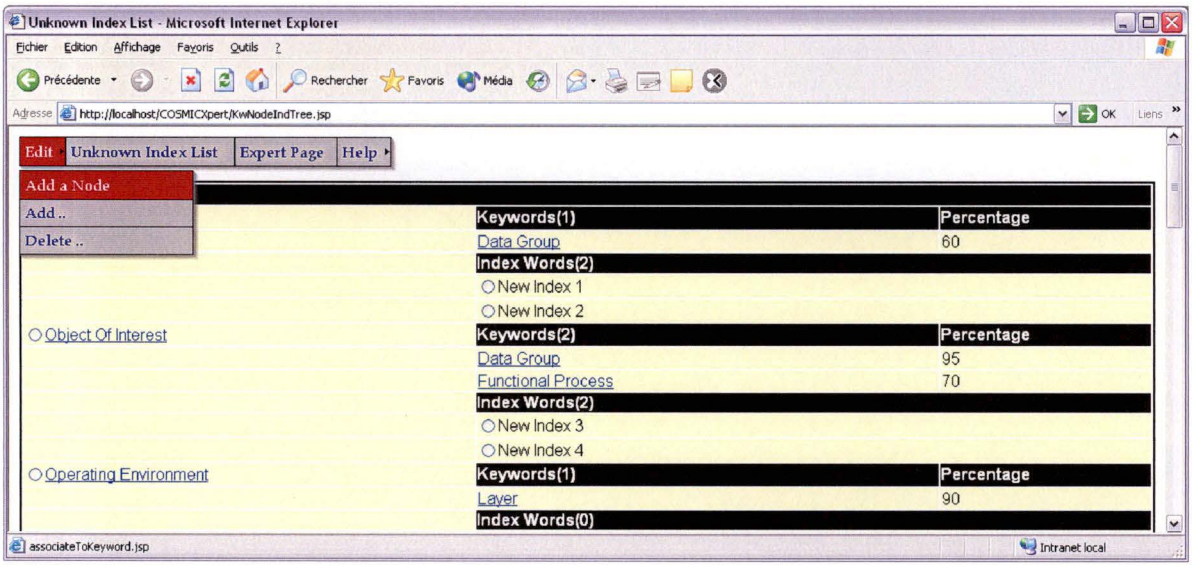


Figure 4.43 *Sélection de la fonction « Ajout d'un nœud »*

Une fois ce choix effectué, l'expert est prié d'introduire diverses informations. En effet, un nœud est composé d'un nom, d'une description et de lien vers un ou plusieurs mots-clés. De plus, les associations entre un nœud et les mots-clés sont munies d'un pourcentage, représentant le niveau de confidence entre les deux concepts.

Les différentes informations requises par le système sont présentées à la Figure 4.44.

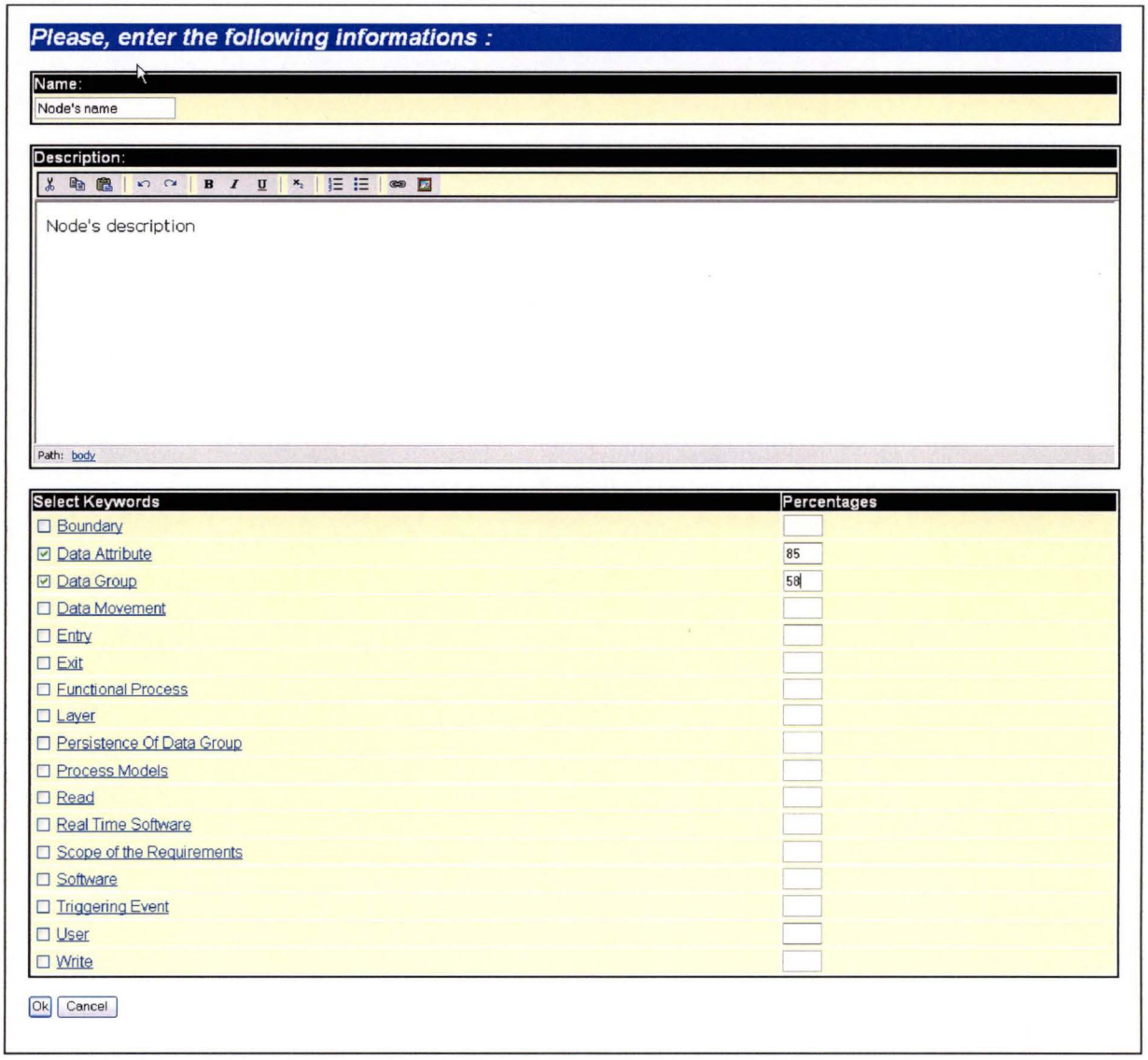


Figure 4.44 *Ajout d'un nœud*

Annexe 5 : Illustration (Ajout d'un Concept Topologique)

Le processus d'ajout d'un concept topologique débute lorsque l'expert sélectionne « *Add a Topological Concept* » sur l'interface qui lui est destinée dans le système, comme le montre la Figure 5.45. Il est composé de neuf étapes, dans lesquelles l'expert est invité à introduire diverses informations. Tous les champs qui sont présentés à l'expert doivent être compléter, à l'exception des champs contenant la mention explicite « *optional* ».

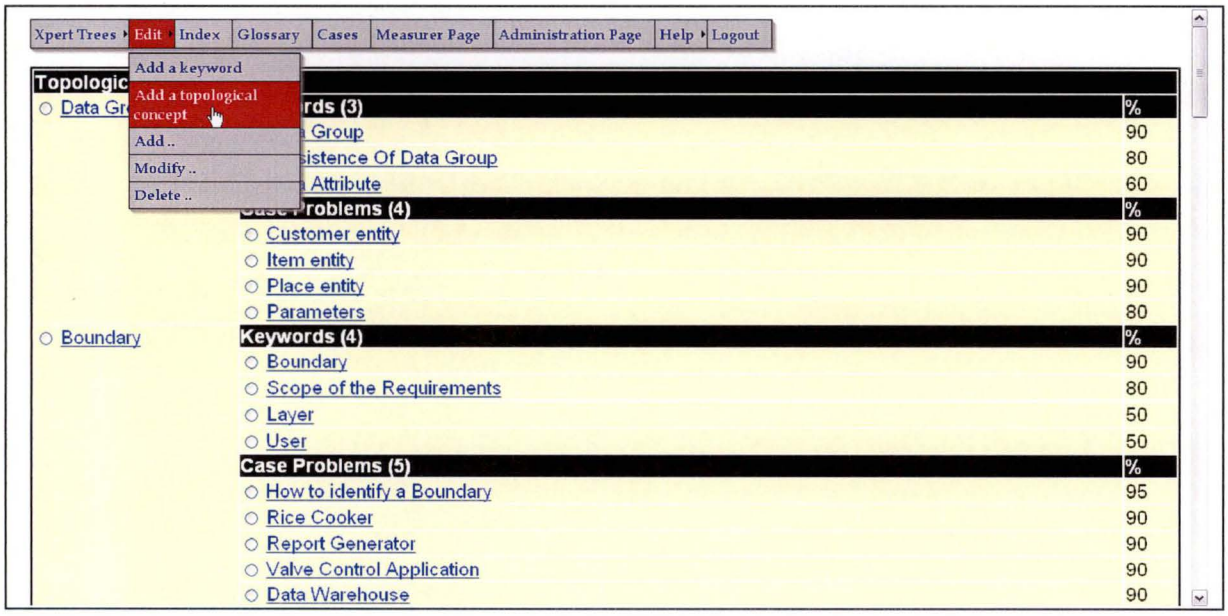


Figure 5.45 *Sélection de la fonction "Ajout d'un concept topologique"*

Le nouveau concept topologique ne sera créé qu'à la fin de toutes ces étapes. Les changements ne seront appliqués dans la base de connaissances qu'une fois que toutes ces étapes auront été complétées.

L'expert est libre de quitter le processus d'ajout d'un concept topologique, ou même le système, quand il le souhaite, sans danger pour la base de connaissances.

Les étapes sont les suivantes :

- Création d'un mot-clé,
- Création du concept topologique,
- Ajout d'une explication (optionnel),
- Ajout d'un thème (première partie),
- Ajout d'un thème (deuxième partie),
- Ajout d'un cas problème (première partie),
- Ajout d'un cas problème (deuxième partie),
- Ajout d'une recommandation au cas problème (troisième partie),
- Confirmation

5.1 Création d'un mot-clé

La création préalable d'un mot-clé est indispensable dans le processus d'ajout d'un concept topologique. Il s'agit en fait de créer un mot-clé qui sera attaché comme étant la définition du nouveau concept topologique. Le mot-clé et le concept topologique seront inséparables par la suite.

L'expert sera en mesure d'associer d'autres mots-clés au nouveau concept topologique par la suite, via l'interface « expert ».

L'expert est prié d'introduire les informations suivantes :

- *Keyword* : le nom du nouveau mot-clé.
- *Description* : la définition du mot-clé. Elle décrit le mot-clé et le concept qu'il représente. Ce champ peut contenir divers type de données, comme du texte formaté et des images.
- *Percentage* : représente la magnitude du lien unissant le mot-clé et le concept topologique

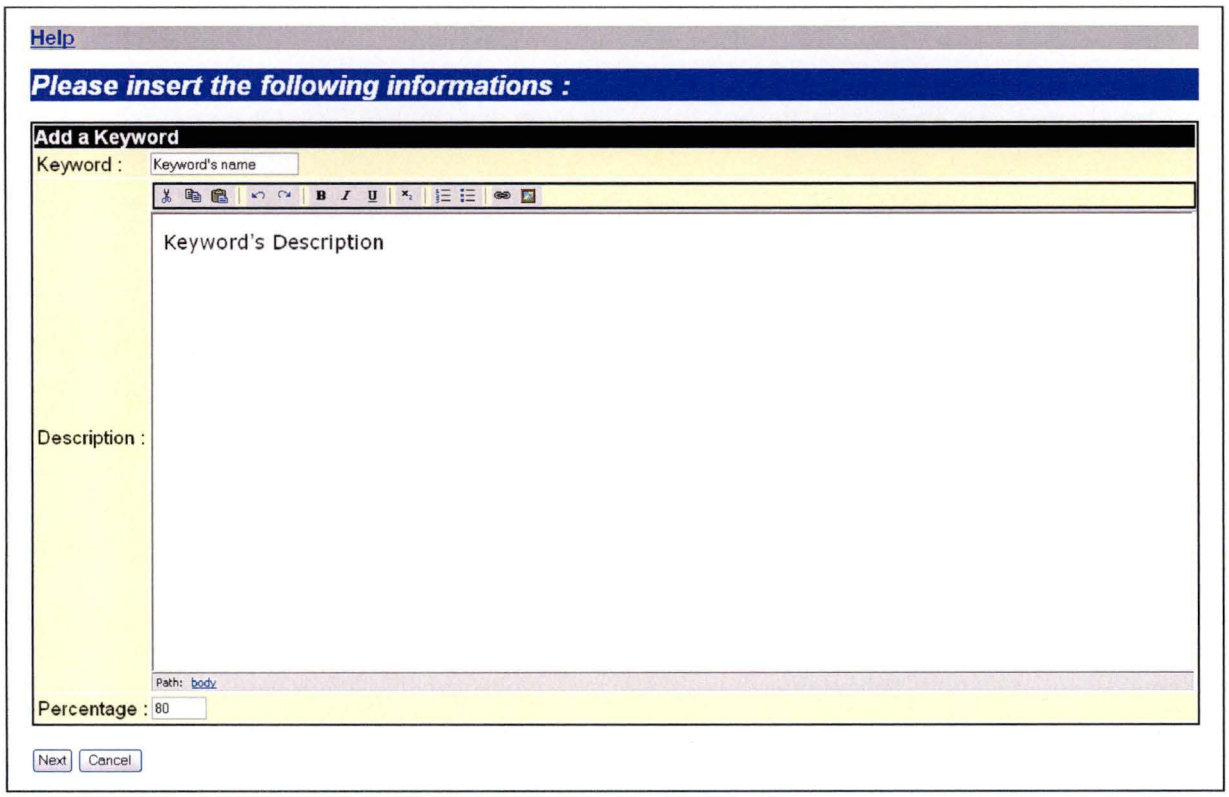


Figure 5.46 *Création d'un nouveau mot-clé*

5.2 Création du concept topologique

Après la première étape, l'expert est en mesure de créer le concept topologique proprement dit. Le système lui demande les informations suivantes :

- *Name* : le nom du nouveau concept topologique
- *Type* : le type du concept topologique. L'expert peut choisir l'un des types proposés. A savoir, « temps réel », « logiciel de gestion d'informations » ou « autre ».
- *Related Keywords and/or Nodes* : représente les liens vers des mots-clés/nœuds, qui vont ajouter des explications à la définition du nouveau concept topologique. Ce champ est optionnel.

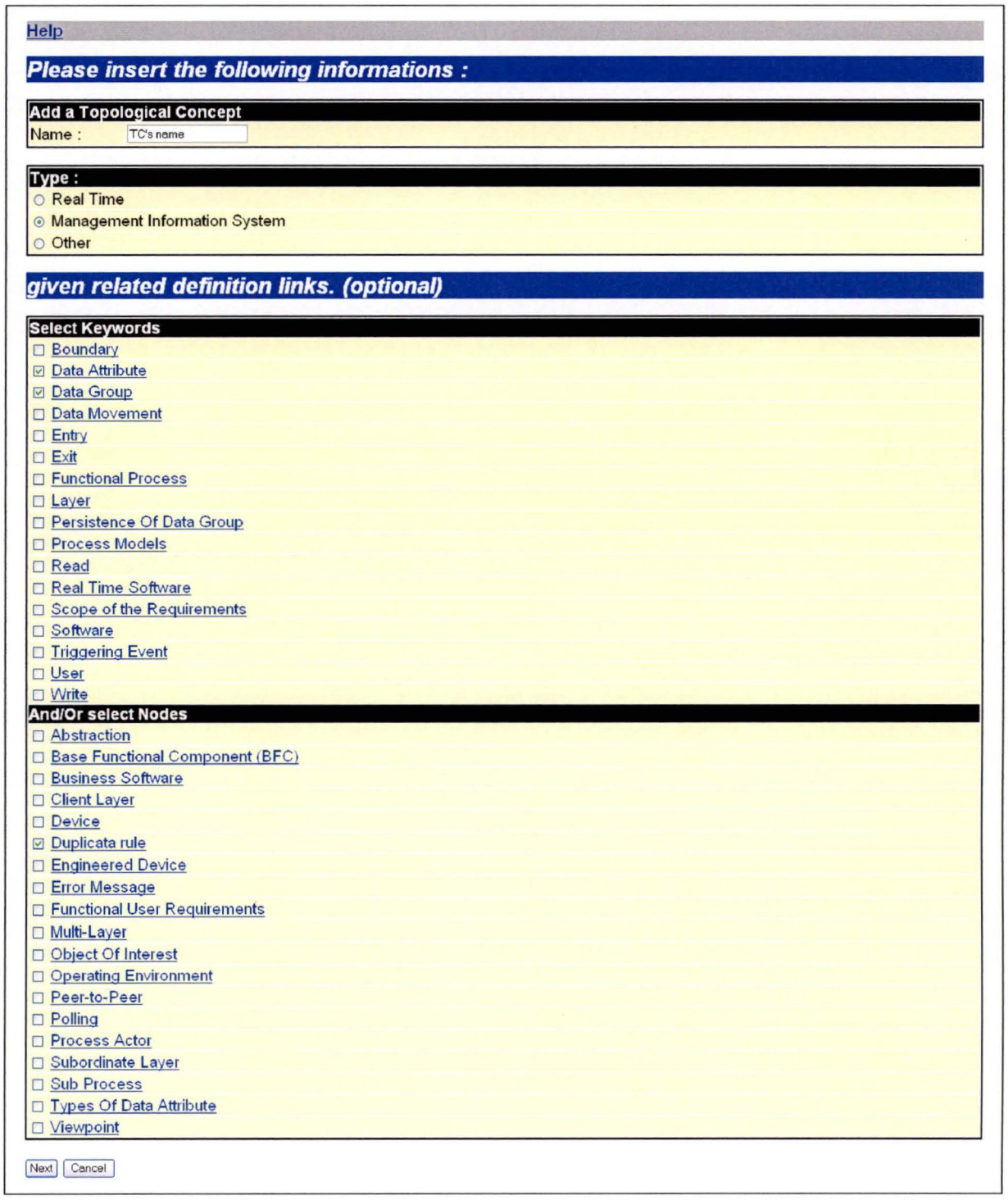


Figure 5.47 *Création du nouveau concept topologique*

5.3 Ajout d'une explication (optionnel)

L'expert a la possibilité d'ajouter une explication supplémentaire au concept topologique qu'il est en train de créer. Cette explication pourra aider les mesureurs à comprendre le sens du concept représenté.

L'expert est en mesure d'ajouter autant d'explications qu'il le souhaite. Celles-ci sont fortement recommandées, mais ne sont pas obligatoires.

L'explication se compose de :

- *Name* : le nom de l'explication.
- *Explanation* : l'explication elle-même. Ce champ peut contenir aussi bien du texte formaté que des images.

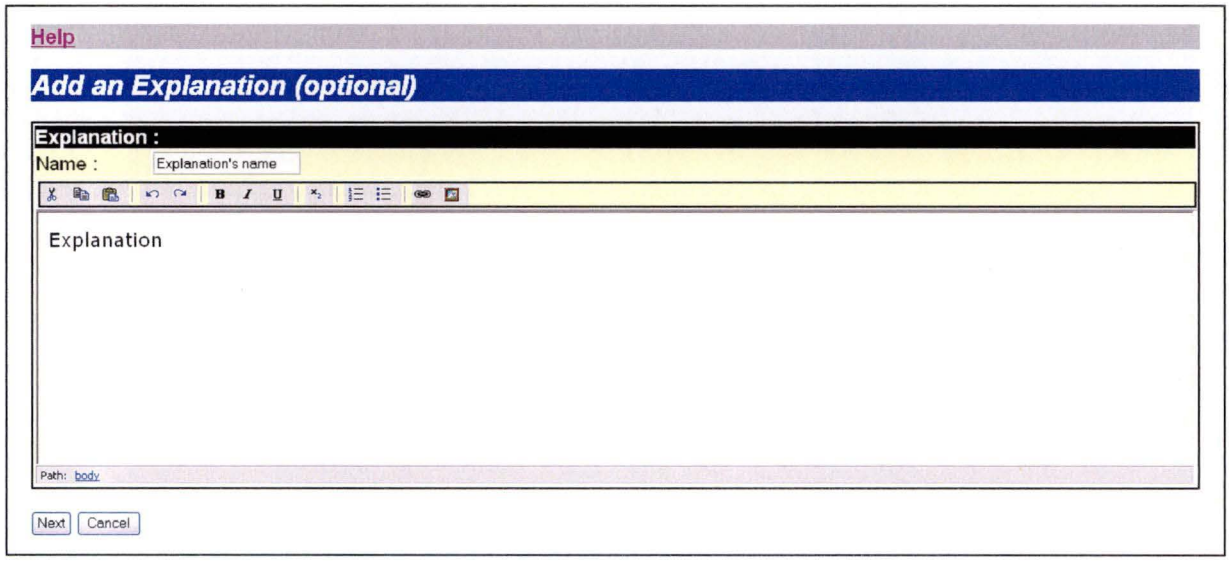


Figure 5.48 *Ajout d'une explication*

5.4 Ajout d'un thème (première partie)

Tout concept topologique doit contenir des thèmes, de façon à ce qu'il puisse être utilisé par les mesureurs. En considérant les concepts topologiques comme des *principes*, on peut assimiler les thèmes à des *règles* associées à ces principes.

L'expert doit introduire au moins un thème mais il a la possibilité d'en introduire autant qu'il le souhaite.

Les informations requises sont :

- *Percentage* : le niveau de confidence du lien entre le concept topologique et ce thème.
- *Question* : la formulation du thème. Il s'agit de la question à laquelle les mesureurs devront répondre pour identifier un concept.
- *Why* : représente le concept de COSMIC-FFP sur lequel est basé ce thème.
- *Facts* : propositions de réponses à la question représentée par ce thème. L'expert doit donner le pourcentage de chaque réponse possible. Ce pourcentage sera utilisé pour calculer quelle recommandation doit être affichée aux mesureurs en fonction de leurs réponses.

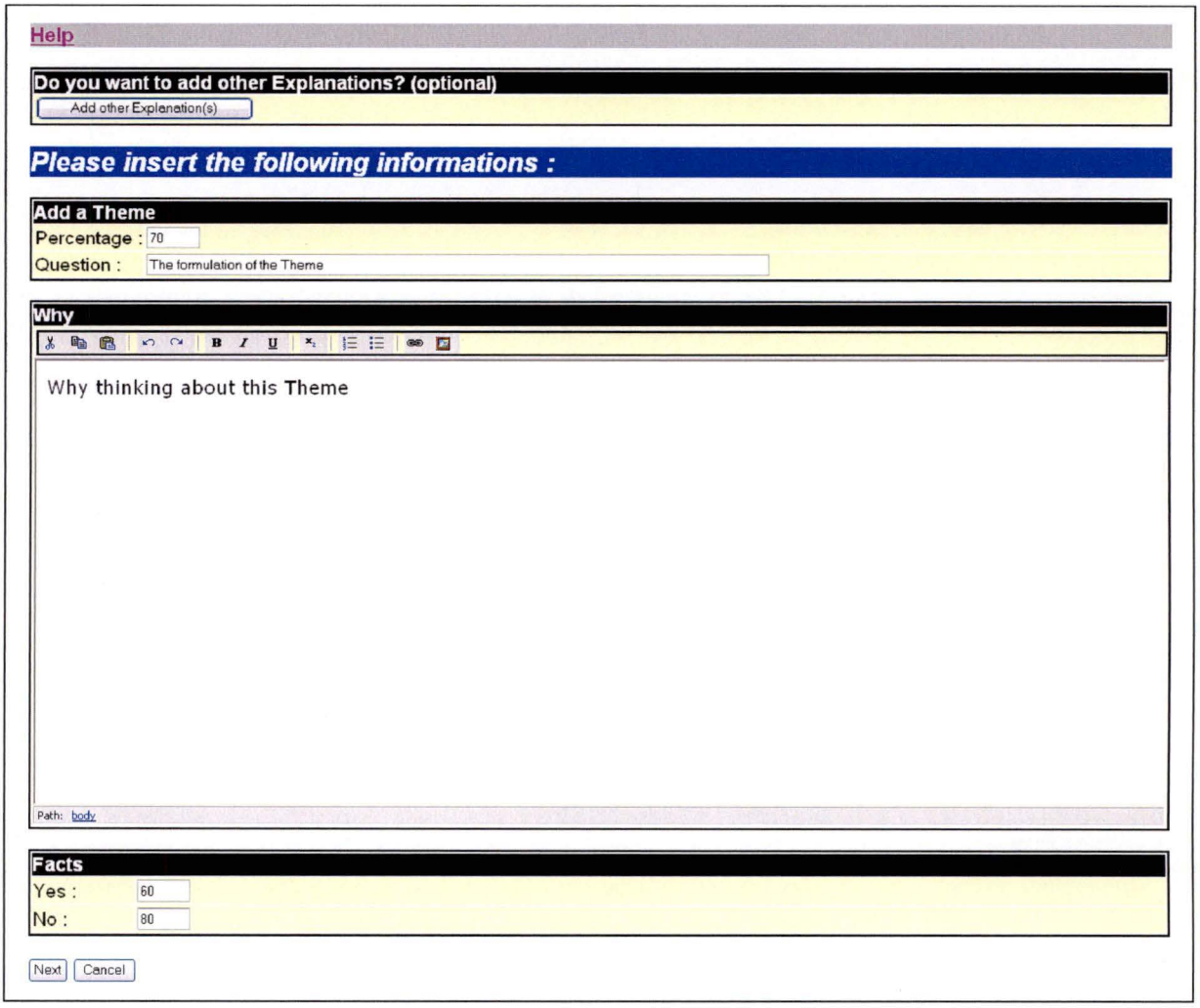


Figure 5.49 *Ajout d'un thème (première partie)*

5.5 Ajout d'un thème (deuxième partie)

Certaines informations d'un thème ne sont pas obligatoires. Cette étape demande à l'expert s'il veut ajouter ces informations facultatives.

Il s'agit de :

- *Rules* : les règles du manuel de COSMIC-FFP associées à ce thème.
- *Related Keywords and/or Nodes* : liens vers d'autres mots-clés/nœuds du glossaire. Ces liens offrent un complément de définition au thème.

Figure 5.50 *Ajout d'un thème (deuxième partie)*

5.6 Ajout d'un cas problème (première partie)

Un concept topologique n'est d'aucune utilité sans thèmes. Il n'est pas plus utile sans cas problèmes.

Un cas problème représente le contexte dans lequel le mesureur doit identifier un concept.

Le système demande à l'expert d'introduire quelques informations, à travers plusieurs étapes, vu la quantité et la diversité des données.

Dans cette première page, les informations requises sont :

- *Case Problem* : le nom du cas problème.
- *Percentage* : le niveau de confidence entre ce cas problème et le concept topologique.
- *Problem identification* : l'explication du cas problème. Elle définit les frontières du cas problème.

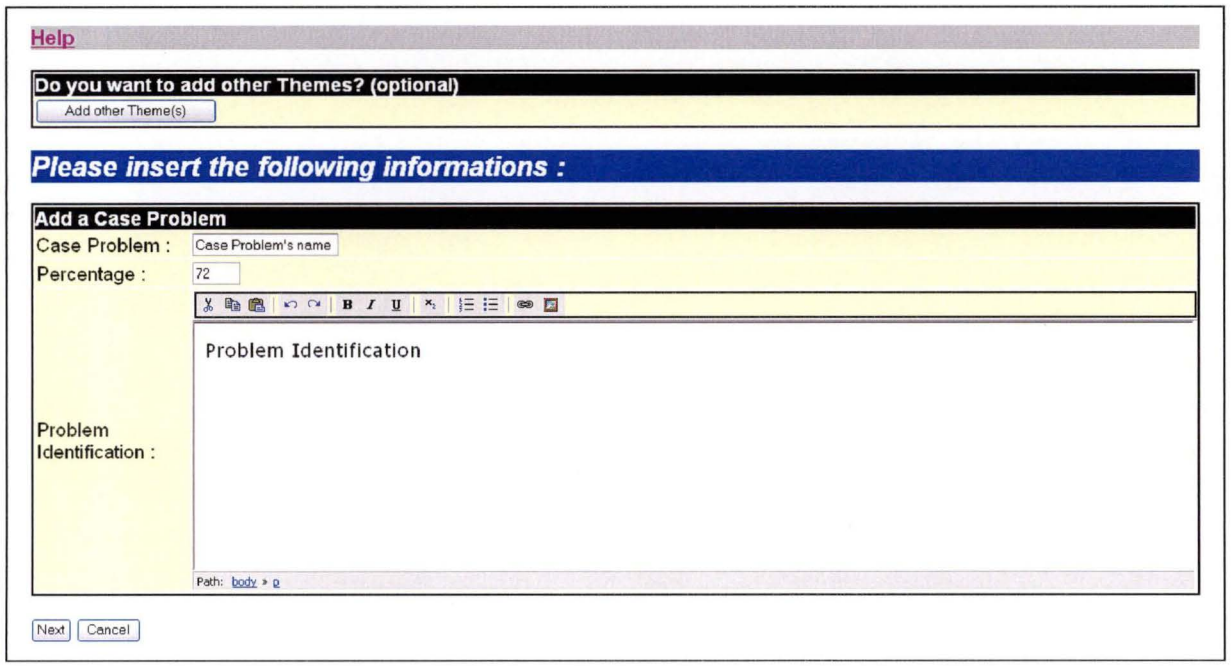


Figure 5.51 *Ajout d'un cas de problème (première partie)*

5.7 Ajout d'un cas problème (deuxième partie)

Dans cette seconde étape de l'ajout d'un cas problème, l'expert doit introduire :

- *Définition* : une explication détaillée du cas problème. L'expert doit exprimer les propres exigences du cas problème. Ces exigences sont reprises dans la documentation du cas d'étude associé ou dans le manuel de COSMIC-FFP pour les problèmes de type « Générique ».

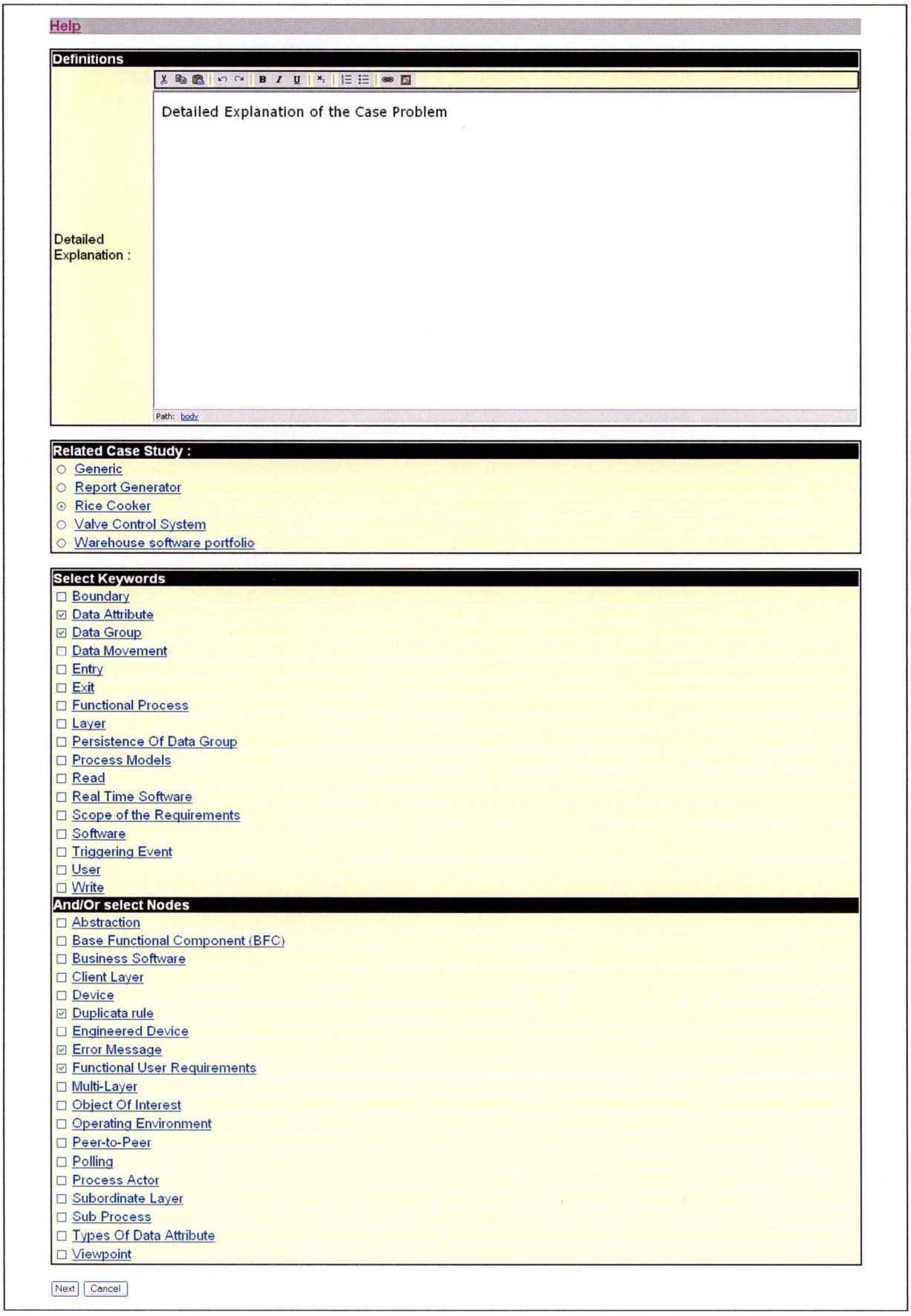


Figure 5.52 *Ajout d'un cas problème (deuxième partie)*

5.8 Ajout d'une recommandation au cas problème (troisième partie)

De façon à ce que les experts puissent identifier les concepts, il est nécessaire de leur proposer des recommandations, en fonction de leurs réponses. Ces recommandations seront ajoutées par l'expert dans cette étape.

Les informations requises pour ajouter une recommandation sont les suivantes :

- *Answer* : la solution qu cas problème, selon le fait choisi.
- *Minimum percentage* : la borne inférieur de la plage de pourcentage dans laquelle la recommandation sera affichée aux mesureurs. Cette borne est calculée automatiquement par le système.
- *Maximum percentage* : la borne supérieure jusqu'à laquelle la recommandation sera affichée.
- *Recommendation* : l'explication du résultat que recevront les mesureurs. Celle-ci lui présente les raisons pour lesquelles il a identifié ou non le concept, selon le contexte.
- *Related Keywords and/or Nodes* : liens vers des mots-clés et/ou nœuds du glossaire.

L'expert doit introduire un minimum de deux recommandations pour chaque cas problème. Le système demandera l'ajout d'une nouvelle recommandation jusqu'au moment où le pourcentage maximum sera égal à 100.

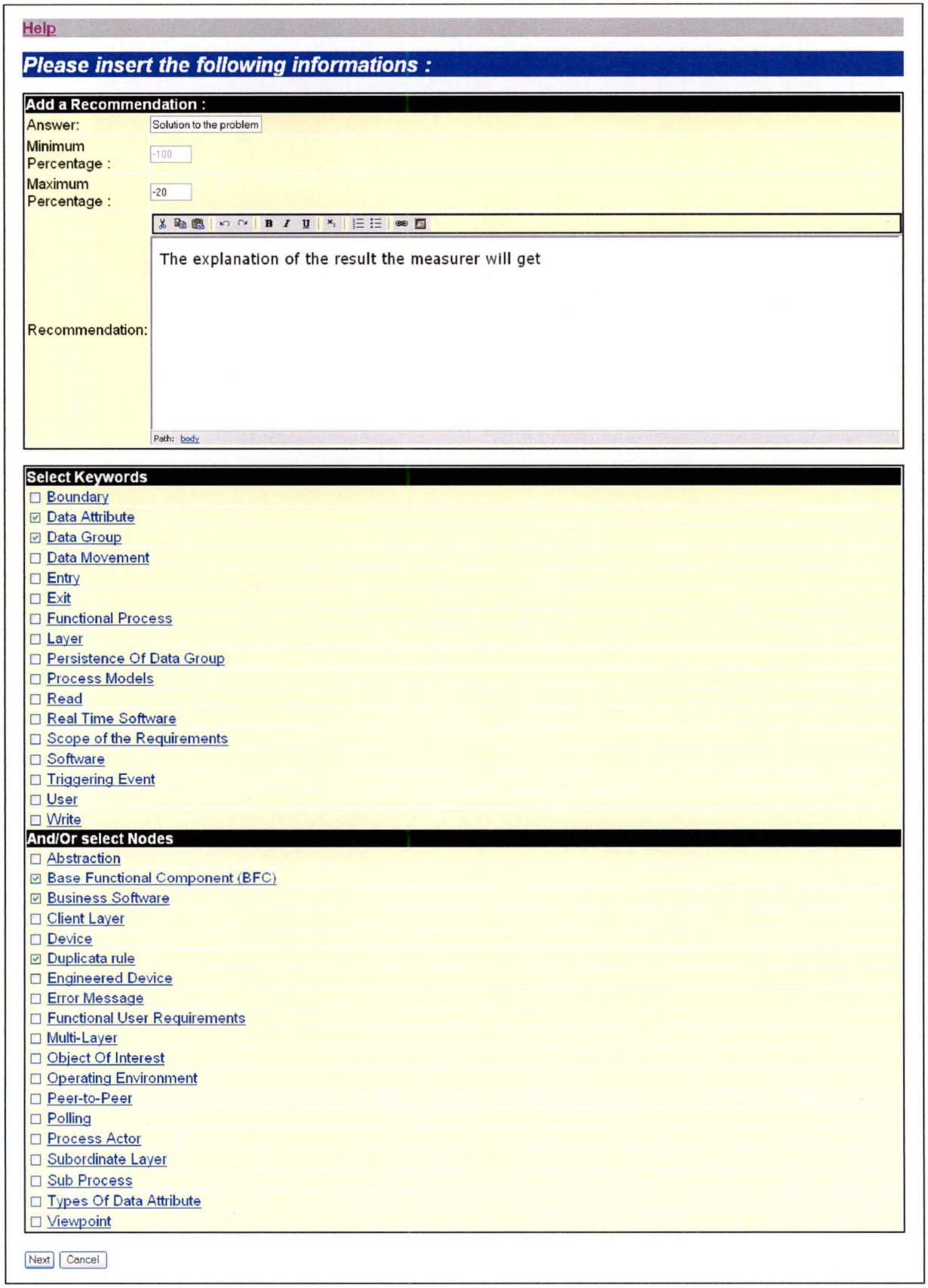


Figure 5.53 *Ajout d'une recommandation au cas problème*

5.9 Confirmation

L'expert peut choisir d'ajouter un autre cas problème, ou de créer le nouveau concept topologique tout de suite.

Le système informe l'expert si le concept a été créé correctement ou si une erreur s'est produite.

Il s'agit de la fin du processus de création d'un concept topologique. L'expert peut alors demander l'exécution d'autres fonctionnalités du système.

Toutes les informations introduites à travers ce processus peuvent être modifiées par la suite, en utilisant les diverses fonctionnalités associées aux différents concepts de la base de connaissances de CosmicXpert.

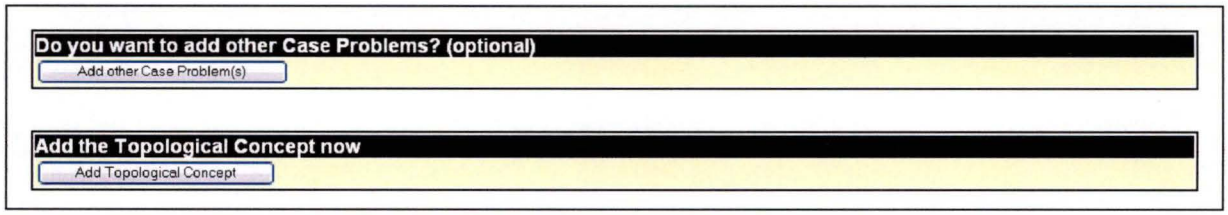


Figure 5.54 *Ajout d'un autre cas problème ou Confirmation de l'ajout du concept topologique*

Bibliographie

[1] ISO/IEC-19761:2003, "Ingénierie du logiciel -- COSMIC-FFP -- Méthode fonctionnelle de la mesure de taille," 2003.

[2] J.-M. Desharnais, "Application de la mesure fonctionnelle COSMIC-FFP: une approche cognitive," in *Informatique*. Montréal: UQAM, 2003.

[3] T. Küssing, "Design and implementation of a diagnostic prototype to support the application of a software measurement method, mémoire de fin d'étude." Nuremberg: Georg-Simon-Ohm Fachhochschule, 2002.

[4] F. Gruselin and J. Vilz, "Vérification et validation d'un système expert pour la mesure fonctionnelle (CosmicXpert)." Namur: Facultés universitaires Notre-Dame de la Paix, 2003.

[5] K. Parsaye, M. Chignell, S. Khoshafian, and H. Wong, *Intelligent Databases : O-O, Deductive Hypermedia Technologies*: John Wiley & Sons, 1989.

[6] F. Heys-Roth, D. A. Waterman, and D. B. Lenat, *Building Expert Systems*: Addisson-Wesley, 1983.

[7] D. A. Waterman, *A Guide To Expert Systems*: Addisson-Wesley, 1986.

[8] P. Jackson, *Introduction to Expert Systems*: Addison-Wesley, 1986.

[9] G. F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Fourth Edition ed: Addison-Wesley, 2002.

[10] J. Kolodner, *Case-based reasoning*: Morgan-Kaufman, 1993.

[11] B. Boehm, "Verifying and Validating software requirements and design specifications.," *IEEE Software*, vol. 1(1), 1984.

[12] T. Bench-Capon, D. Castelli, F. Coenen, L. Devendeville-Brisoux, B. Eaglestone, N. Fiddian, A. Gray, A. Ligeza, and A. Vermesan, "Report on the 1st International Workshop on Validation, Verification and Integrity Issues of Expert and Database Systems.," presented at 1st International Workshop on Validation, Verification and Integrity Issues of Expert and Database Systems, 1999.

[13] A. Preece, "Evaluating Verification and Validation Methods in Knowledge Engineering," in *Industrial Knowledge Management, A Micro-Level Approach*, I. R. R. (ed), Ed.: Springer, 2001, pp. 123-145.

[14] R. Nouira and J.-M. Fouet, "A Knowledge Based Tool for Checking Large Knowledge Bases," *Journal du Laboratoire des Systèmes d'Information (43 Boulevard du 11 Novembre 1918 69622 Villeurbanne).*

[15] D. S. Spreeuwenberg, I. R. Gerrits, and D. M. Boekenoogen, "VALENS : A Knowledge Based Tool to Validate and Verify an Aion Knowledge Base."

[16] S. Zahran, *Software Process Improvement. Practical Guidelines for Business Success*, 1998.

[17] A. Preece, "Towards a Quality Assessment Framework for Knowledge-Based Systems," *Journal of Systems and Software*, vol. 29(3), 1995.

[18] M. Mattei, "Verification, Validation et Test dans le cycle de développement des systèmes à base de connaissances," *EDF : Collection de notes internes de la Direction des Etudes et Recherches : Mathématiques, informatique, télécommunications*, vol. 92 NI J 0002, pp. 18, 1991.

[19] V. S. Mookerjee and M. V. Mannino, "Sequential Decision Models for Expert System Optimization," *IEEE Software*, vol. 9, pp. 675-687, 1997.
[20] M. Ayel and J.-P. Laurent, "Design, development and validation of expert systems : a survey of developers," in *Verification , Validation and Test of Knowledge-Based Systems.*, Eds., Ed. New York: John Wiley & Sons, 1991, pp. 3-20.
[21] R. Knauf, A. J. Gonzalez, and T. Abel, " A Framework for Validation of Rule-Based Systems," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 32(3), pp. 281-295, 2002.
[22] J.-L. Hainaut, "DB-MAIN, Ingénierie des bases de données, matières approfondies." Namur, 2000, pp. 3.2-5.26.
[23] J.-L. Hainaut, "Ingénierie des bases de données." Namur, 2002, pp. 2.2-2.6.
[24] J.-M. Desharnais, A. Abran, A. Mayers, L. Buglione, and V. Bevo, "A Knowledge-Based System in Functional Size Measurement," 2002.
[25] J.-M. Desharnais and A. Abran, "Applying a Functional Measurement Method: Cognitive Issues," presented at Internal Workshop on Software Measurment (IWSM '01), 2001.
[26] N. Habra and S. Alexandre, "Méthodologie de Développement Logiciel (matière approfondie)." Namur, 2003.
[27] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile softaware developpment methods - Review and analysis*. Espoo: VTT Technical Research of Finland - VTT Electronics, 2002.
[28] ISO/IEC-12207:1995, "Technologies -- Processus du cycle de vie du logiciel," 1995.
[29] D. Leffingwell and D. Widrig, *Managing Software Requirements, A Use Case Approach*, Second ed. Boston: Addison-Wesley, 2003.
[30] K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn, W. Cunningham, M. Flower, J. Greening, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marrick, R. C. Martin, S. Mellor, K. Shwaber, J. Sutherland, and D. Thomas, "Software Manifesto," http://www.agilemanifesto.org, 2001.
[31] A. Cockburn, "Crystal Clear : A Human-Powered Methodology For Small Teams," http://alistair.cockburn.us/crystal/index.html, 2004.